Split the quads along their shortest diagonal.

Longest Diagonal

Split the quads along their longest diagonal. This is the preferred mode for cloth simulations.

N-gon Method

Beauty

Arrange the new triangles nicely, slower method.

Clip

Splits n-gons using an ear-clipping algorithm (the same method of tessellation used for viewport display).

Minimum Vertices

Minimum number of vertices a face must have to be triangulated. For example, setting this value to 5, will prevent triangulation of Quads and only triangulate N-gons.

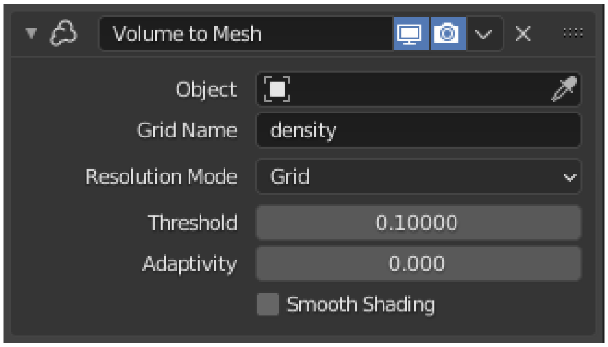
Volume to Mesh Modifier

This modifier is the inverse of the *Mesh to Volume* modifier. It takes an existing volume object and converts one of its grids to a mesh. Only scalar grids (such as the density grid) can be converted.

Tip

To copy and move the generated mesh separately from the volume object, use a collection instance.

Options



Object

Grid Name: density

Resolution Mode: Grid

Threshold: 0.10000

Adaptivity: 0.000

☐ Smooth Shading

The Volume to Mesh modifier.

Object

The source volume object.

Grid Name

The name of the grid that will be converted. This has to be a scalar grid.

Resolution Mode

Mode for how the resolution of the final mesh is controlled.

Grid

This makes the resolution dependent on the resolution of the grid that is converted. Higher resolution grids result in a higher resolution mesh. In many cases, that is the most efficient mode.

Voxel Amount

Specifies the approximate resolution of the final mesh. The voxel size is adapted to the size of the entire volume.

Voxel Size

Use a fixed resolution that does not change when the volume changes.

Threshold

Voxels with a larger value are considered to be inside the mesh and all other voxels outside. The mesh will be generated on the boundary of inside and outside voxels. This is sometimes also called the “iso value”.

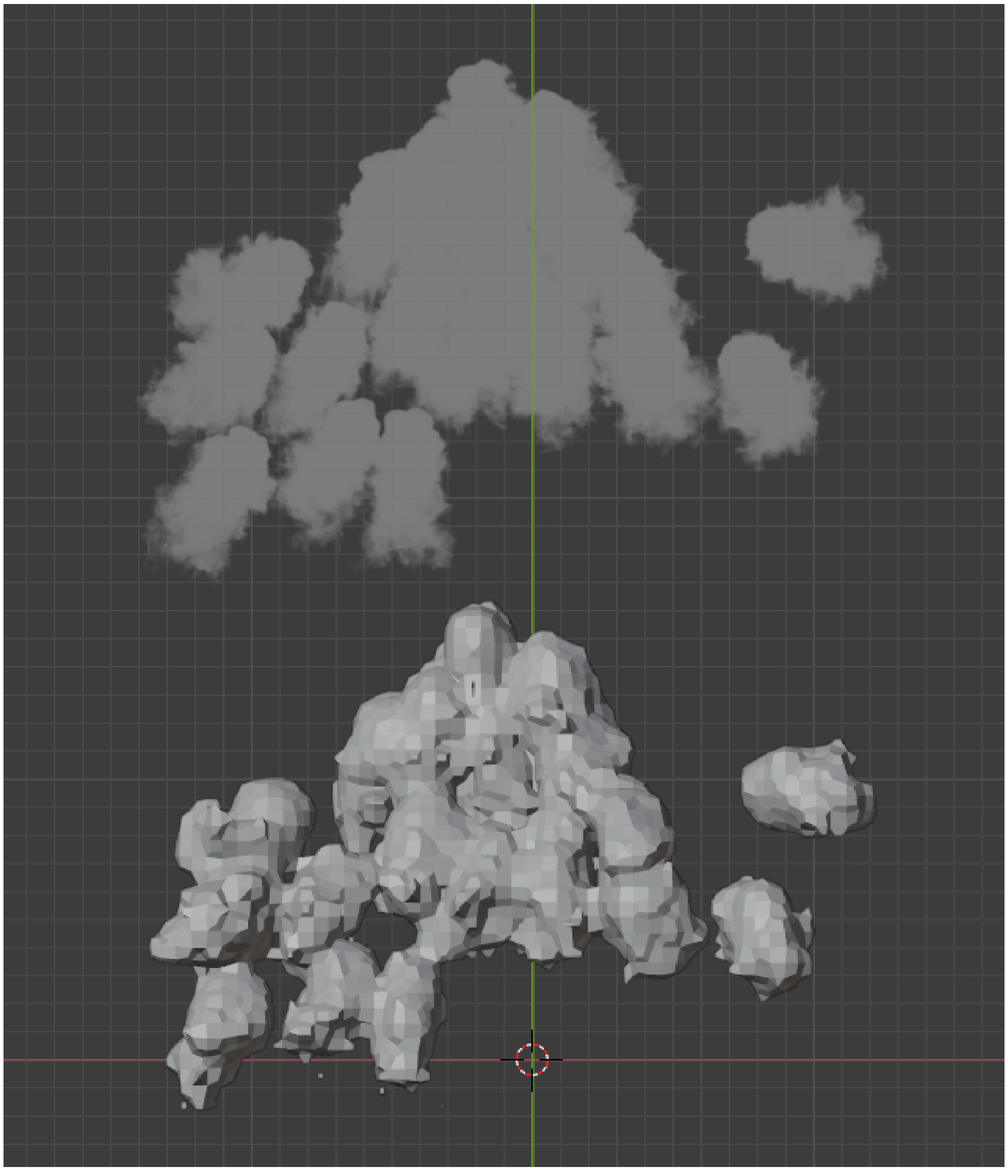
Adaptivity

This is similar to decimating the final to reduce resolution where it is not needed.

Smooth Shading

Enables smooth shading on the generated mesh.

Example

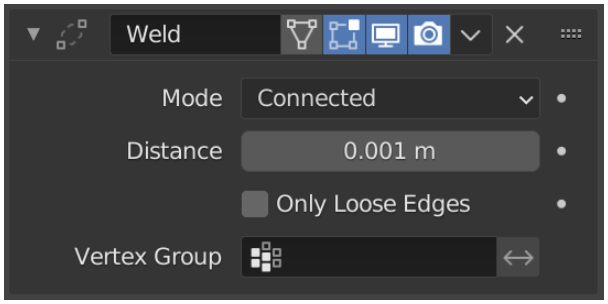


Converting a cloud-shaped volume to a mesh.

Weld Modifier

The *Weld* modifier looks for groups of vertices within a threshold and merges them, collapsing the surrounding geometry.

Options



Mode: Connected

Distance: 0.001 m

☐ Only Loose Edges

Vertex Group:

The Weld modifier.

Mode

Method for choosing which vertices are merged.

All::	Merge includes all geometry including loose parts.
Connected::	Merge only includes attached geometry i.e. the modifier will not merge loose parts together.

Distance

Maximum distance that the vertices must have each other to be merged.

Only Loose Edges *Connected Mode*

Only collapse short edges which are not adjacent to any face. This is useful for example to stitch the seams used in cloth simulations.

Vertex Group

When the *Vertex Group* option is selected, only vertices with weight above zero will be affected by the modifier.

Invert <–>

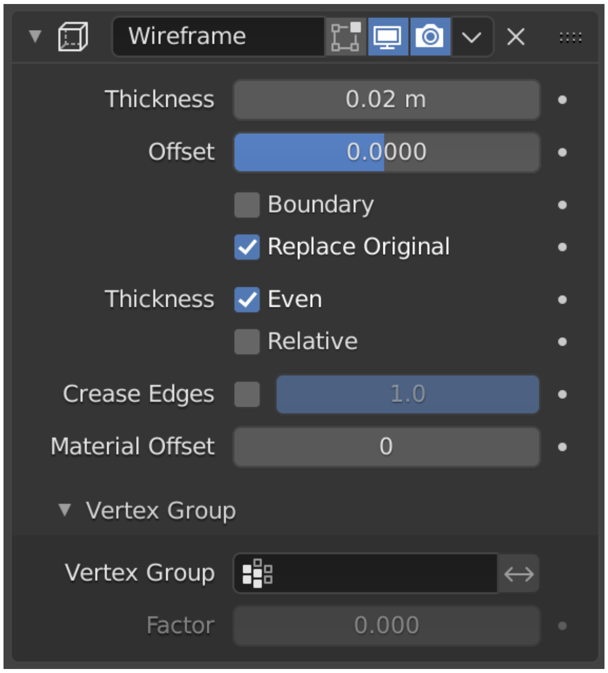
Inverts the influence of the selected vertex group, meaning that the group now represents vertices that will not be merged by the modifier.

The setting reverses the weight values of the group.

Wireframe Modifier

The *Wireframe* modifier transforms a mesh into a wireframe by iterating over its faces, collecting all edges and turning those edges into four-sided polygons. Be aware of the fact that your mesh needs to have faces to be wireframed. You can define the thickness, the material and several other parameters of the generated wireframe dynamically via the given modifier options.

Options



Thickness 0.02 m
Offset 0.0000
☐ Boundary
☑ Replace Original
Thickness ☑ Even
☐ Relative
Crease Edges ☐ 1.0
Material Offset 0
Vertex Group
Factor 0.000

The Wireframe modifier.

Thickness

> The depth or size of the wireframes.

Offset

> A value between (-1 to 1) to change whether the wireframes are generated inside or outside of the original mesh. Set to zero, *Offset* will center the wireframes around the original edges.

Boundary

> Creates wireframes on mesh island boundaries.

Replace Original

> If this option is enabled, the original mesh is replaced by the generated wireframe. If not, the wireframe is generated on top of it.

Thickness

Even

Maintain thickness by adjusting for sharp corners. Sometimes improves quality but also increases computation time.

Relative

Determines the edge thickness by the length of the edge. Longer edges will be thicker.

Crease Edges

This option is intended for usage with the Subdivision modifier. Enable this option to crease edges on their junctions and prevent large curved intersections.

Crease Weight

Define how much crease (0 to 1, nothing to full) the junctions should receive.

Material Offset

Uses the chosen material index as the material for the wireframe; this is applied as an offset from the first material.

Warning

Wireframe thickness is an approximation. While *Even Thickness* should yield good results in many cases, skinny faces can cause ugly spikes. In this case you can either reduce the extreme angles in the geometry or disable the *Even Thickness* option.

Vertex Group

The weights of the selected vertex group are multiplied onto the *Thickness*, so vertices with lower weights will be less thick. The vertices which are not part of the vertex group will be used as if their weight was zero.

Invert

Reverses the vertex group weights, so that the used weight is one minus the actual weight.

Factor

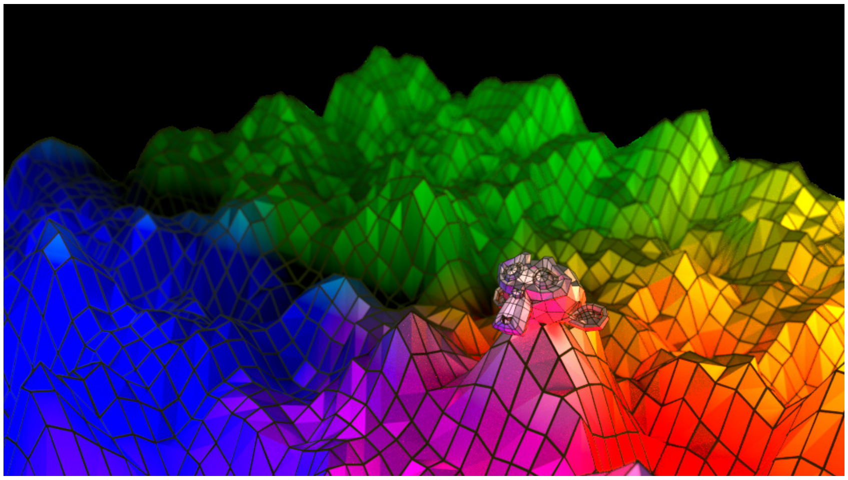
How much the vertex weights are taken into account.

- On 0.0 , vertices with zero weight will have no thickness at all.
- On 0.5 , vertices with zero weight will be half as thick as those with full weight.
- On 1.0 , the weights are ignored and the *Thickness* value is used for every vertex.

Note

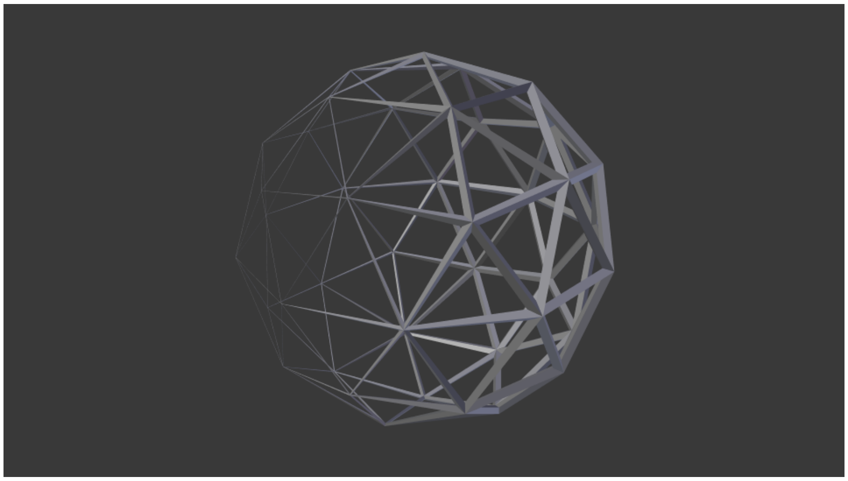
If the final thickness of a vertex is zero, it will still generate a wireframe. Therefore creating duplicate geometry, which sometimes needs extra care.

Examples



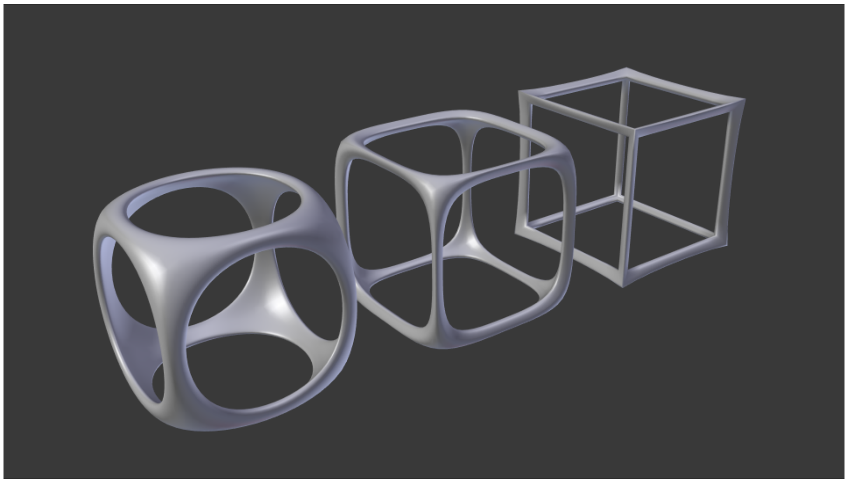
Wireframes on a displaced plane.

In this example, the wireframes carry a second (dark) material while the displaced plane uses its original one.



Vertex group weighting.

The weights of the vertex group gradually change from 0 to 1.



Wireframe and Subdivision Surface modifier.

Cube with enabled *Crease Edges* option. The *Crease Weight* is set to 0, 0.5 and 1.

Deform

- Armature Modifier
- Cast Modifier
- Curve Modifier
- Displace Modifier
- Hook Modifier
- Laplacian Deform Modifier
- Lattice Modifier
- Mesh Deform Modifier
- Shrinkwrap Modifier
- Simple Deform Modifier
- Smooth Modifier
- Smooth Corrective Modifier
- Smooth Laplacian Modifier
- Surface Deform Modifier
- Volume Displace Modifier
- Warp Modifier
- Wave Modifier

Armature Modifier

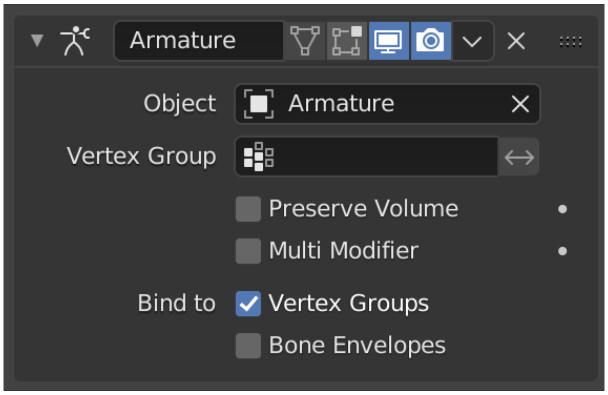
The *Armature* modifier is used for building skeletal systems (rigs) for animating the poses of characters and anything else which needs to be posed.

By adding an armature system to an object, that object can be deformed accurately so that geometry does not have to be animated by hand.

See also

For more details on armatures usage, see the armature section.

Options



Object ☐ Armature

Vertex Group

☐ Preserve Volume

☐ Multi Modifier

Bind to ☑ Vertex Groups

☐ Bone Envelopes

The Armature modifier.

Object

The name of the armature object used by this modifier.

Vertex Group

A vertex group of the object, which weights will be used to determine the influence of this modifier's results when mixing it with the results from other *Armature* ones.

Only meaningful when having at least two of these modifiers on the same object, with *Multi Modifier* activated.

Invert <–>

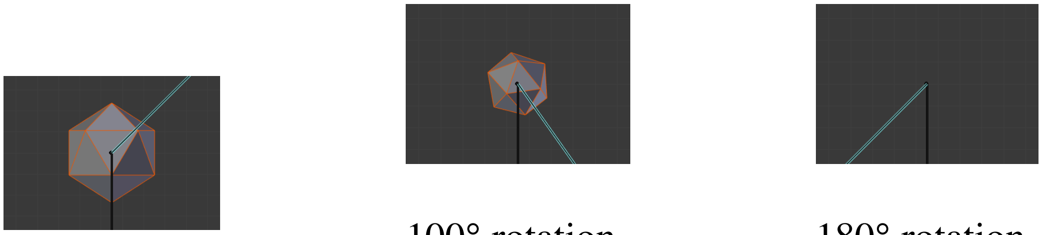
Inverts the influence set by the vertex group defined in previous setting (i.e. reverses the weight values of this group).

Preserve Volume

Use quaternions for preserving volume of object during deformation. It can be better in many situations.

Without it, rotations at joints tend to scale down the neighboring geometry, up to nearly zero at 180 degrees from rest position. With it, the geometry is no longer scaled down, but there is a “gap”, a discontinuity when reaching 180 degrees from rest position.

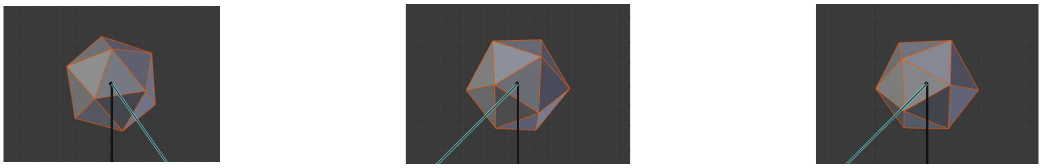
Example of *Preserve Volume* effects. Note that the icosphere is deformed using the envelopes weights.



Initial state.

100° rotation, *Preserve Volume* disabled.

180° rotation, *Preserve Volume* disabled.



100° rotation, *Preserve Volume* enabled.

179.9° rotation, *Preserve Volume* enabled.

180.1° rotation, *Preserve Volume* enabled.

Multi Modifier

Use the same data as a previous modifier (usually also an *Armature* one) as input. This allows you to use several armatures to deform the same object, all based on the "non-deformed" data (i.e. this avoids having the second *Armature* modifier deform the result of the first one…).

The results of the *Armature* modifiers are then mixed together, using the weights of the *Vertex Group* as "mixing guides".

> **Tip**
>
> *Armature* modifiers can quickly be added to objects by parenting them to an armature.

Bind to

Methods to bind the armature to the mesh.

Vertex Groups

Meshes and lattices only. When enabled, bones of a given name will deform vertices which belong to vertex groups of the same name. E.g. a bone named "forearm", will only affect the vertices in the "forearm" vertex group.

The influence of one bone on a given vertex is controlled by the weight of this vertex in the relevant group. A much more precise method than *Bone Envelopes*, but also generally longer to set up.

Bone Envelopes

When enabled, bones will deform vertices or control points near them, defined by each bone's envelope radius and distance. This lets bone envelopes control the deformation (i.e. bones deform vertices in their neighborhood).

Example of skinning methods.

Cast Modifier

The *Cast* modifier shifts the shape of a mesh, curve, surface or lattice, towards any of a few predefined shapes (sphere, cylinder, cuboid).

It is equivalent to the To Sphere tool in Edit Mode, and what other programs call "Spherify" or "Spherize", but, as written above, it is not limited to casting to a sphere.

Tip

The Smooth Modifier is a good companion to *Cast*, since the cast shape sometimes needs smoothing to look nicer or even to fix shading artifacts.

Note

For performance reasons, this modifier only works with local coordinates. If the modified object looks wrong, you may need to apply its transformations, especially when casting to a cylinder.

Options

Shape

Menu to choose target shape of the projection: *Sphere*, *Cylinder* or *Cuboid*.

Axis

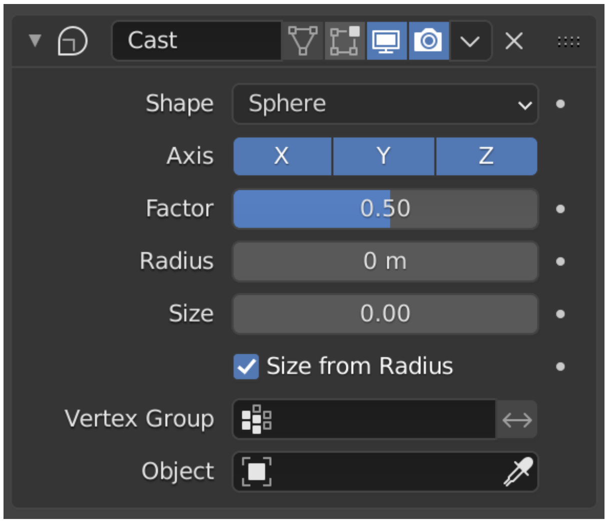
Toggle buttons to enable/disable the modifier in the X, Y, Z axes directions (X and Y only for *Cylinder* cast type, since the Z axis remains unaffected).

Factor

The factor to control blending between original and cast vertex positions.

It is a linear interpolation: 0.0 gives original coordinates (i.e. modifier has no effect), 1.0 casts to the target shape.

Values below 0.0 or above 1.0 exaggerate the deformation, sometimes in interesting ways.



Cast

Shape: Sphere

Axis: X Y Z

Factor: 0.50

Radius: 0 m

Size: 0.00

☑ Size from Radius

Vertex Group

Object

The Cast modifier.

Radius

If nonzero, this radius defines a sphere of influence. Vertices outside it are not affected by the modifier.

Size

Alternative size for the projected shape. If zero, it is defined by the initial shape and the control object, if any.

Size from Radius

If activated, calculate *Size* from *Radius*, for smoother results.

Vertex Group

If set, restrict the effect to the only vertices in that vertex group. This allows selective, real-time casting, by painting vertex weights.

Invert <–>

Inverts the influence of the selected vertex group, meaning that the group now represents vertices that will not be deformed by the modifier.

The setting reverses the weight values of the group.

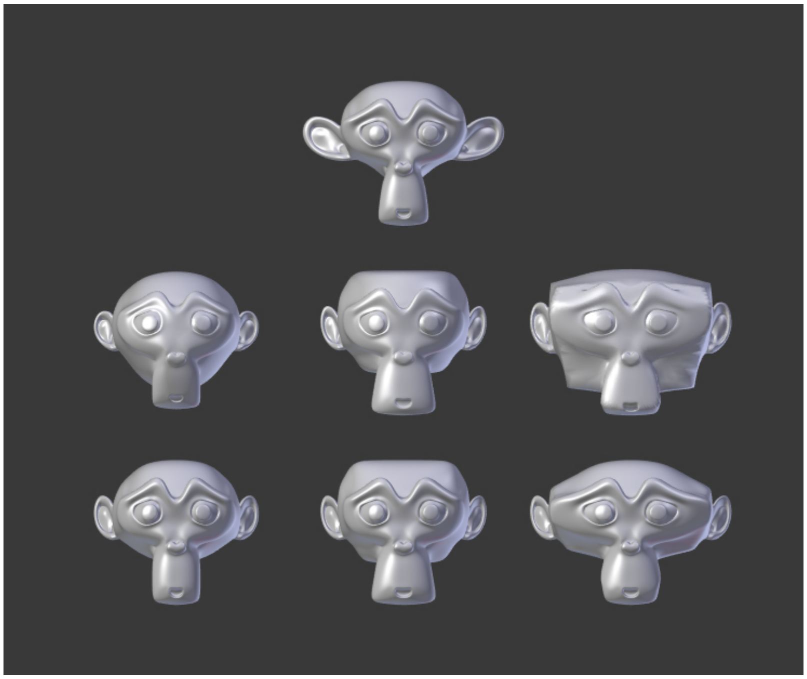
Object

The name of an object to control the effect. The location of this object's origin defines the center of the projection. Also, its size and rotation transform the projected vertices.

Hint

Animating (keyframing) this control object also animates the modified object's casting deformation.

Example



Top: Suzanne without modifiers. Middle: Suzanne with each type of Cast Modifier (Sphere, Cylinder and Cuboid). Bottom: Same as above, but now only X axis is enabled. Sample blend-file.

Curve Modifier

The *Curve* modifier provides a simple but efficient method of deforming a mesh along a curve object.

It works on a (global) dominant axis, X, Y, or Z. This means that when you move your mesh in the dominant direction (by default, the X axis), the mesh will traverse along the curve, as if it was a train following and deforming along rails. Moving the mesh perpendicularly to this axis, the object will move closer or further away from the curve.

When you move the object beyond the curve's ends, the object will continue to deform based on the direction vector at those ends.

Note

This modifier works in global space, in other words, the actual position of the geometry relative to the curve is determinant to get a correct result.

Typically, you'll want your object's origin to be at the center of your geometry (not offset far away from it, you can e.g. Set Origin to Geometry).

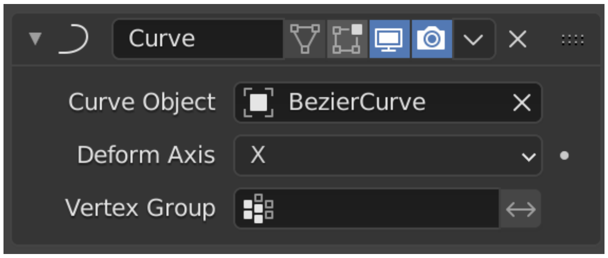
And then you'll want to start with your object's origin at the same location as your curve object's origin (you may use snap tools for that…).

If the curve is 3D, the *Tilt* value of its control points will be used to twist the deformed object. And the *Radius* property controls the size of the object as well. Those options are in the *Shape* panel, under Path/Curve-Deform.

Options

Curve Object

The name of the curve object that will affect the deformed object.


Curve

Curve Object: BezierCurve

Deform Axis: X

Vertex Group:

The Curve modifier.

Deformation Axis

This is the axis that the curve deforms along.

X/Y/Z/-X/-Y/-Z

Vertex Group

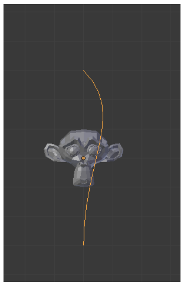
If set, restrict the effect to the only vertices in that vertex group.

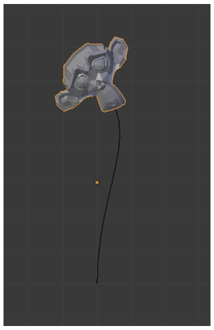
Invert <–>

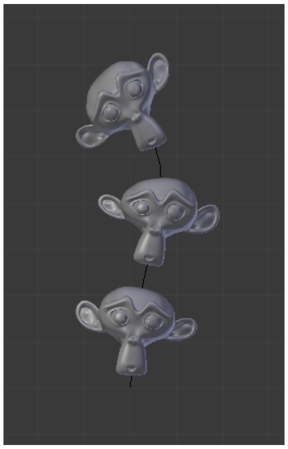
Inverts the influence of the selected vertex group, meaning that the group now represents vertices that will not be deformed by the modifier.

The setting reverses the weight values of the group.

Example


Edit curve.


Monkey on a curve.

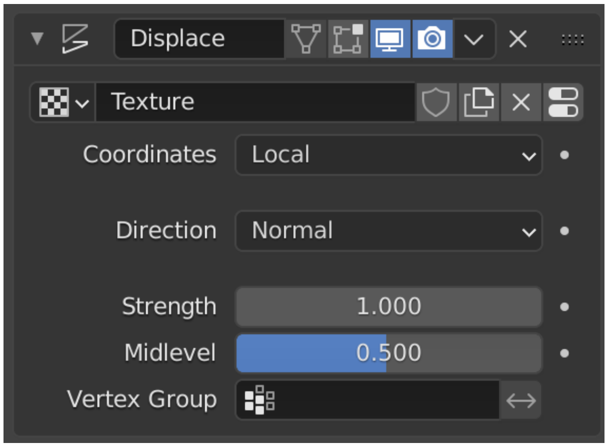

Monkey deformations.

Displace Modifier

The *Displace* modifier displaces vertices in a mesh based on the intensity of a texture. Either procedural or image textures can be used.

The displacement can be along a particular local axis, along the vertex normal, or the separate RGB components of the texture can be used to displace vertices in the local X, Y and Z directions simultaneously (sometimes referred to as *Vector Displacement*).

Options



Coordinates: Local

Direction: Normal

Strength: 1.000

Midlevel: 0.500

Vertex Group

The Displace modifier.

Texture

The name of the texture from which the displacement for each vertex is derived. If this field is empty, the modifier defaults to 1.0 (white).

Coordinates

The texture coordinate system to use when retrieving values from the texture for each vertex.

See common masking options for a complete reference.

Direction

The direction along which to displace the vertices. Can be one of the following:

X, Y, Z

Displace along an axis.

Normal

Displace along the vertex normal.

Custom Normal

Displace along (averaged) custom normals, instead of vertex normals.

RGB to XYZ

Displace along local XYZ axes individually using the RGB components of the texture (Red values displaced along the X axis, Green along the Y, Blue along the Z).

Space

With a direction set to X, Y, Z, or XYZ the modifier can either displace along local or global axes.

Strength

The strength of the displacement. After offsetting by the *Midlevel* value, the displacement will be multiplied by the *Strength* value to give the final vertex offset.

$vertex\_offset = displacement \times Strength$

A negative strength can be used to invert the effect of the modifier.

Midlevel

The texture value which will be treated as no displacement by the modifier. Texture values below this threshold will result in negative displacement along the selected direction, while texture values above it will result in positive displacement.

$displacement = texture\_value - Midlevel$

Recall that color/luminosity values are typically between (0.0 to 1.0) in Blender, and not between (0 to 255).

Vertex Group

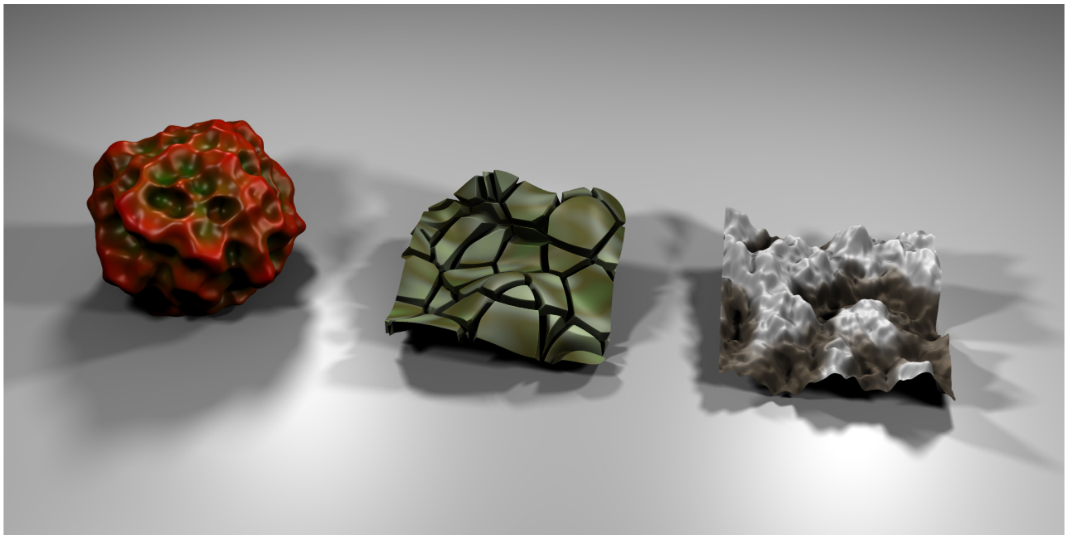
The name of a vertex group which is used to control the influence of the modifier. If left empty, the modifier affects all vertices equally.

Invert <–>

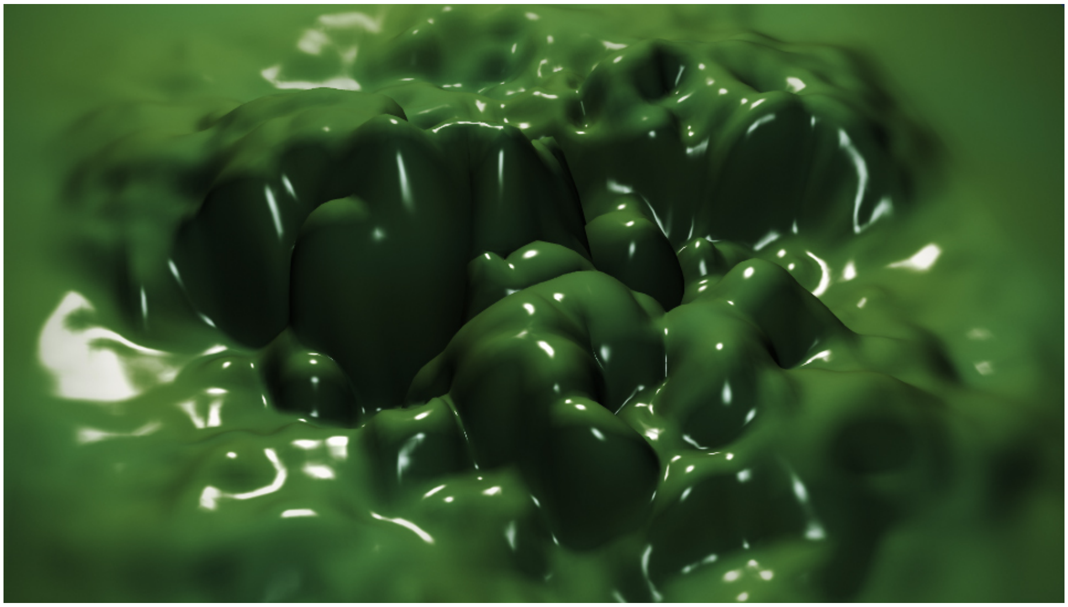
Inverts the influence of the selected vertex group, meaning that the group now represents vertices that will not be deformed by the modifier.

The setting reverses the weight values of the group.

Example



Three different objects created with the Displace modifier. Sample blend-file.



A slime animation created with the Displace modifier. [Sample blend-file](#).

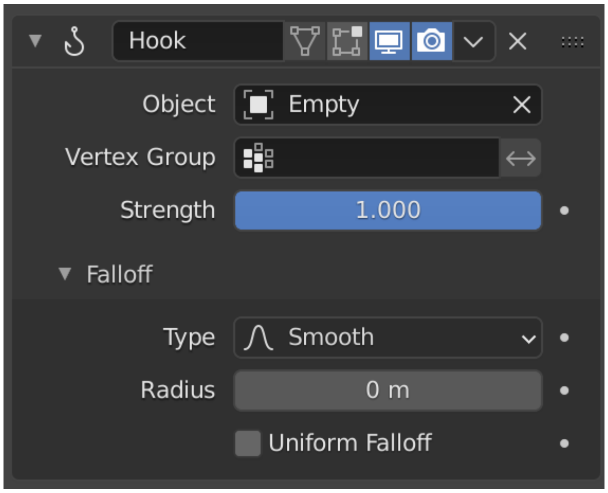
Hook Modifier

The *Hook* modifier is used to deform a mesh, curve or lattice using another object (usually an empty or a bone but it can be any object). As an object specified as hook moves, it pulls vertices or control points from the geometry with it. You can think of it as animated Proportional Editing.

While hooks do not give you the fine control over vertices movement that shape keys do, they have the advantage that you can select vertices directly for manipulation.

To assign selected vertices (in Edit Mode) you can use the *Assign* button on the modifier panel or use the Add Hook menu.

Options


Hook

Object: Empty

Vertex Group:

Strength: 1.000

Type: Smooth

Radius: 0 m

☐ Uniform Falloff

The Hook modifier.

Object

The name of the object to hook vertices to.

Vertex Group

Allows you to define the influence per vertex.

Useful when you want something other than a spherical field of influence.

Invert <–>

Inverts the influence of the selected vertex group, meaning that the group now represents vertices that will not be deformed by the modifier.

The setting reverses the weight values of the group.

Strength
Adjust this hooks influence on the vertices, were (0.0 to 1.0) (no change to fully follows the hook).

Since multiple hooks can work on the same vertices, you can weight the influence of a hook using this property.

The following settings are only available in Edit Mode:

Reset
Recalculate and clear the offset transform of the hook.

Recenter
Set the hook center to the 3D cursor position.

Select
Select the vertices affected by this hook.

Assign
Assigns selected vertices to this hook.

Warning

The Hook Modifier stores vertex indices from the original mesh to determine what to affect. This means that modifiers that generate geometry, like *Subdivision Surface*, should always be put **after** the *Hook* one in the stack. Otherwise, the generated geometry will be left out of the hook's influence.

Falloff

Type
This can be used to adjust the kind of influence curve that the hook has on the mesh. You can also define a custom curve to get a much higher level of control.

Radius

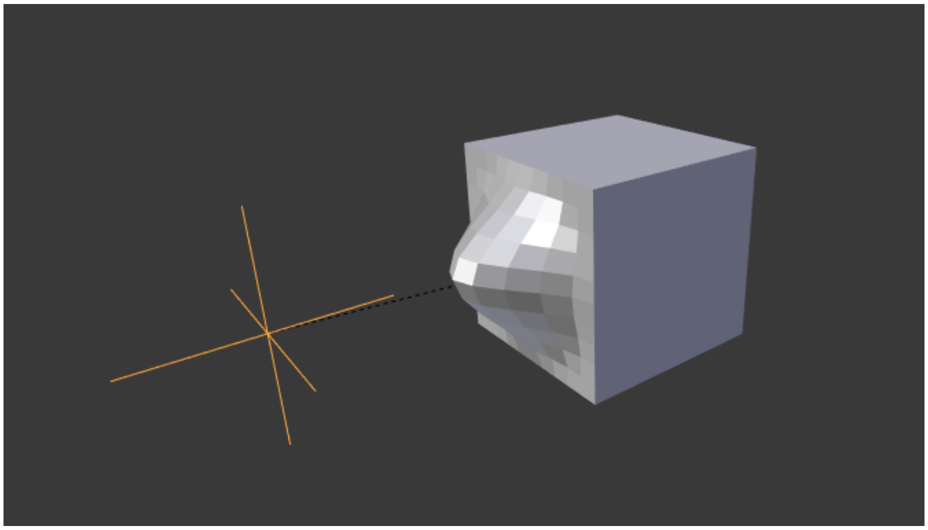
The size of the hooks influence.

Uniform Falloff

This setting is useful when using hooks on scaled objects, especially in cases where non-uniform scale would stretch the result of the hook.

This is especially useful for lattices, where it is common to use non-uniform scaling.

Example



Empty used as Hook to deform a subdivided cube.

Laplacian Deform Modifier

The *Laplacian Deform* modifier allows you to pose a mesh while preserving geometric details of the surface.

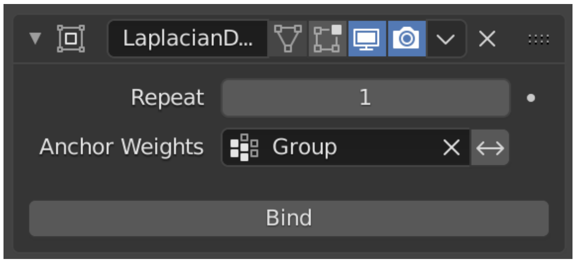
The user defines a set of “anchor” vertices, and then moves some of them around. The modifier keeps the rest of the anchor vertices in fixed positions and calculates the optimal locations of all the remaining vertices to preserve the original geometric details.

This modifier captures the geometric details with the use of differential coordinates. The differential coordinates capture the local geometric information, the curvature and direction of a vertex based on its neighbors.

Note

You must define an *Anchors Vertex Group*. Without it the modifier does nothing.

Options



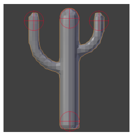
Repeat: 1

Anchor Weights: Group

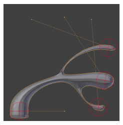
The Laplacian Deform modifier.

Repeat
  How many iterations to do to improve the found solution. The objective is to find the rotation of the differential coordinates preserving the best possible geometric details. Details are retained better if more iterations are used, however, it will take longer to calculate.

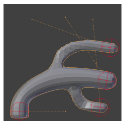
Deform horse example blend-file.



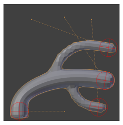
Original Model.



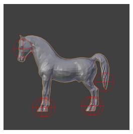
Repeat: 1.



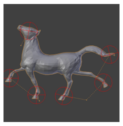
Repeat: 2.



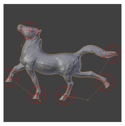
Repeat: 5.



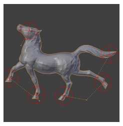
Original Model.



Repeat: 1.



Repeat: 2.



Repeat: 10.

Anchor Weights

The group of vertices that the user will use to transform the model. The weight of each vertex does not affect the behavior of the modifier, the method only takes into account vertices with weight greater than 0.

Invert <–>

Inverts the influence of the selected vertex group, meaning that the group now represents vertices that will not be deformed by the modifier.

The setting reverses the weight values of the group.

Bind

The *Bind* button is what tells the *Laplacian Deform* modifier to actually capture the geometry details of the object, so that altering the anchor vertices actually alters the shape of the deformed object.

Unbind

After binding the modifier, you may later decide to make changes to the *Anchors Vertex Group*. To do so you will first need to *Unbind* the modifier before binding it again.

Error Messages

Vertex group *group_name* is not valid

This message is displayed when a user deletes the vertex group or changes its name.

Vertices changed from X to Y

This message is displayed when a user adds or deletes vertices to/from the mesh.

Edges changed from X to Y

This message is displayed when a user adds or deletes edges to/from the mesh.

The system did not find a solution

This message is displayed if the solver could not find a solution for the linear system.

Note

If the mesh is dense, with a number of vertices greater than 100,000, then it is possible that the nonlinear optimization system will fail.

History

Laplacian Surface Editing is a method developed by Olga Sorkine and others in 2004. This method preserves geometric details as much as possible while the user makes editing operations. This method uses differential coordinates corresponding to the difference between a vector and the weighted average of its neighbors to represent the local geometric detail of the mesh.

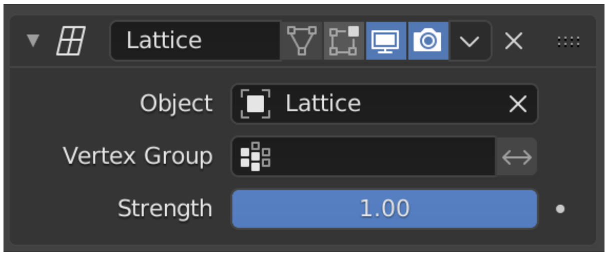
Lattice Modifier

The *Lattice* modifier deforms the base object according to the shape of a Lattice object. Objects to be deformed can be meshes, curves, surfaces, text, lattices and even particles.

Tip

A *Lattice* modifier can quickly be added to selected objects by parenting them using the *Lattice Deform* option.

Options



Lattice

Object: Lattice

Vertex Group:

Strength: 1.00

The Lattice modifier.

Object

> The Lattice object with which to deform the base object.

Vertex Group

> An optional vertex group name which lets you limit the modifier's effect to a part of the base mesh.
>
> Invert <–>
>
> > Inverts the influence of the selected vertex group, meaning that the group now represents vertices that will not be deformed by the modifier.
> >
> > The setting reverses the weight values of the group.

Strength

> A factor to control blending between original and deformed vertex positions.

Hints

Why would you use a lattice to deform a mesh instead of deforming the mesh itself in Edit Mode? There are a couple of reasons for that:

- If your object has a large number of vertices, it would be difficult to edit portions of it quickly in Edit Mode. Using a lattice will allow you to deform large portions efficiently.
- The smooth deformation you get from a *Lattice* modifier can be hard to achieve manually.
- Multiple objects can use the same lattice, thus allowing you to edit multiple objects at once.
- Like all modifiers, it is non-destructive. Meaning all changes happen on top of the original geometry, which you can still go back to and edit without affecting the deformation.
- A lattice does not affect the texture coordinates of a mesh's surface.

Note

When using a lattice to deform particles, order in the modifier stack matters. You need to place the *Lattice* modifier after the *Particle System* one.

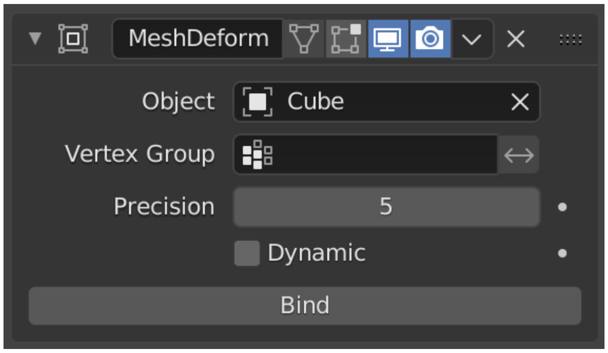
Mesh Deform Modifier

The *Mesh Deform* modifier allows an arbitrary mesh (of any closed shape) to act as a deformation cage around another mesh.

Note

This modifier is reasonably easy to use, but it can be very slow to compute the binding (the mapping between the deform mesh cage to the deformed object geometry).

Options



Object Cube
Vertex Group
Precision 5
☐ Dynamic

The Mesh Deform modifier.

Object
: The name of the mesh object to be used as the deforming cage.

Vertex Group
: An optional vertex group of the object's mesh to restrict the vertices that will be affected by this modifier. Vertices not in this group will not be deformed.

  Invert <–>
  : Inverts the influence of the selected vertex group, meaning that the group now represents vertices that will not be deformed by the modifier.

    The setting reverses the weight values of the group.

Precision

Controls the accuracy with which the deform mesh cage alters the deformed object, when the points on the cage are moved. Raising this value higher can greatly increase the time it takes to complete the binding calculations, but it will get more accurate cage mapping to the deformed object.

This setting becomes unavailable once a cage has been bound.

Dynamic

When activated, other mesh altering features (such as other modifiers and shape keys) are taken into account when binding, increasing deformation quality.

The setting is deactivated by default to save memory and processing time when binding. Like with *Precision*, this setting is unavailable once a cage has been bound.

Bind

Links the current vertex positions of both the modified geometry and the deforming *Object* chosen together. An unbound *Mesh Deform* modifier has no effect, it must be bound so that altering the shape of the deform mesh cage actually alters the shape of the modified object.

Warning

Depending on the settings of the modifier and complexity of the deform mesh cage and/or deformed object, it can take a long time for this operation to complete. This can result in Blender not responding to user's actions until it has completed.

It is also possible that Blender will run out of memory and crash.

To be safe, save your blend-file before proceeding!

Unbind

When a deformed object has been associated to a deform mesh cage, it can later be disassociated by clicking the *Unbind* button which replaced the *Bind* one.

When *Unbind* is clicked, the *deforming mesh cage* will keep its current shape, it will not reset itself back to its initial shape. If you need this original shape, you will have to save a copy of it before you alter it.

The deformed object will, however, reset back to its original shape that it had before it was bound to the deform mesh cage.

Warning

Significant changes to the entire cage mesh *(such as rotating the cage upside down)* can cause noticeable artifacts.

These can be reduced by binding with a higher *Precision*, however, it is a known limitation with this modifier and cannot be avoided entirely.

Hints

- Ensure that the normals on the cage mesh point to the outside (they are used to determine the inside and outside of the cage).
- Besides the outer cage, more faces within the cage, either loose or forming another smaller cage, can be used for extra control. Such smaller cages may also overlap with the main cage. For example, to get extra control over eyes, two small sphere cages could be added around them.

See also

- The Lattice modifier.
- Original paper

Shrinkwrap Modifier

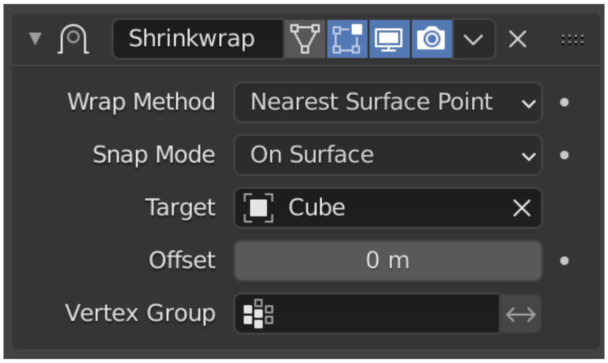
The *Shrinkwrap* modifier allows an object to “shrink” to the surface of another object. It moves each vertex of the object being modified to the closest position on the surface of the given mesh (using one of the four methods available).

It can be applied to meshes, lattices, curves, surfaces and texts.

See also

Shrinkwrap Constraint.

Options



Wrap Method: Nearest Surface Point

Snap Mode: On Surface

Target: Cube

Offset: 0 m

Vertex Group

The Shrinkwrap modifier in Nearest Surface Point mode.

Wrap Method

This selector specifies the method to be used to determine the nearest point on the target’s surface for each vertex of the modified object. Some options will add some extra, specific controls to the panel. See Wrap Methods for an explanation of each method.

Snap Mode

Most modes support an additional setting to control how the vertex is moved to the target point selected by the methods described above. Some of the choices only differ if *Offset* is not zero.

On Surface
: The vertex is always moved. The offset is applied along the projection line connecting the original vertex and selected target point towards the original position.

Outside Surface
: Like *On Surface*, but the offset is always applied towards the outside of the target.

Above Surface
: Like *On Surface*, but the offset is applied along the smooth normal of the target.

Inside
: The vertex is not moved if it is already inside the target. Offset shrinks the allowed volume towards the inside along the projection line.

Outside
: The vertex is not moved if it is already outside the target. Offset expands the exclusion volume towards the outside along the projection line.

Note

The *Inside* and *Outside* options can be used for very crude collision detection. The inside vs outside determination is done based on the target normal and is not always stable near 90 degree and sharper angles in the target mesh.

Target
: Shrink target, the mesh to shrink to/wrap around.

Offset

The distance that must be kept from the calculated target position.

Vertex Group

The vertex group to control whether and how much each vertex is displaced to its target position. If a vertex is not a member of this group, it is not displaced (same as weight 0).

Wrap Methods

Nearest Surface Point

This will select the nearest point over the surface of the shrunk target.

Project

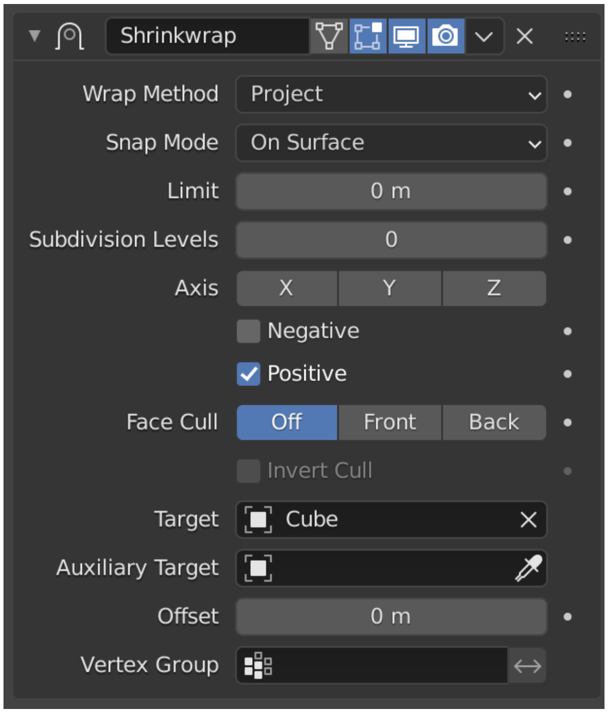
This will project vertices along a chosen axis until they touch the shrink target. Vertices that never touch the shrink target are left in their original position.

Limit

This is a distance limit between original vertex and surface. If the distance is larger than this limit vertex would not be projected onto the surface.

Subdivision Levels

This applies a (temporary) *Catmull-Clark* subdivision to the modified object's geometry, before computing the wrap.



Shrinkwrap

Wrap Method: Project

Snap Mode: On Surface

Limit: 0 m

Subdivision Levels: 0

Axis: X Y Z

☐ Negative

☑ Positive

Face Cull: Off Front Back

☐ Invert Cull

Target: Cube

Auxiliary Target

Offset: 0 m

Vertex Group

Project mode.

Axis

Along which local axis of the modified object the projection is done. These options can be combined with each other, yielding a "median axis” of projection. If none are selected, the normal direction is used.

Negative/Positive

This allows you to select the allowed direction(s) of the shrink along the selected axis. If both options are enabled, both ways are evaluated and the closest hit is selected.

Face Cull

Allows you to prevent any projection over the “front side” (respectively the “back side”) of the target’s faces. The “side” of a face is determined by its normal (front being the side “from where” the normal “originates”).

Invert Cull

If *Cull Faces* is enabled, and *Negative* direction along axis is allowed, this option can be used to invert the *Front* or *Back* cull choice for the *Negative* direction. This is useful when projecting in both directions.

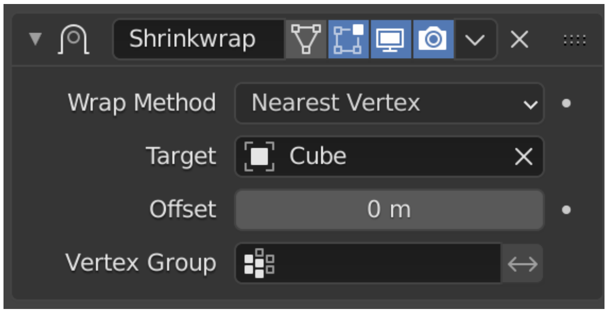
Auxiliary Target

An additional object to project over.

Nearest Vertex

This will snap vertices to the nearest vertex of the shrunk target. It adds no extra options.

This method doesn’t support the *Snap Mode* setting.



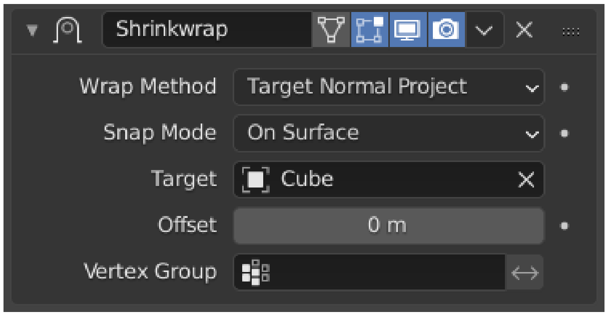
Wrap Method: Nearest Vertex
Target: Cube
Offset: 0 m
Vertex Group:

Nearest Vertex mode.

Target Normal Project

This mode is similar to *Nearest Surface Point*, but produces a much smoother projection in return for being significantly slower.

Instead of finding the closest point, it searches for the nearest point that has its interpolated smooth normal pointing towards or away from the original vertex position. Non-manifold boundary edges are specially handled as infinitely thin cylinders that emit normals in all perpendicular directions; ignores flat shading.

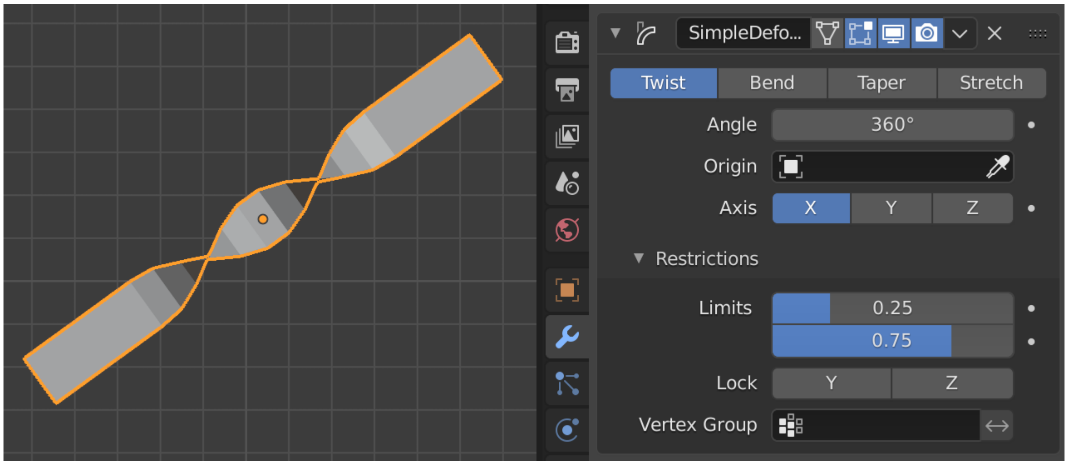


Wrap Method: Target Normal Project
Snap Mode: On Surface
Target: Cube
Offset: 0 m
Vertex Group:

Target Normal Project mode.

Simple Deform Modifier

The *Simple Deform* modifier allows the application of a simple deformation to an object. Meshes, lattices, curves, surfaces and texts are supported objects. The deformation is either a rotation (Twist, Bend) or a scaling (Taper, Stretch). The amount of deformation is specified by the Deform Angle (rotation) or Deform Factor (scaling).



Angle 360°
Origin
Axis X
Limits 0.25
0.75
Lock Y Z
Vertex Group

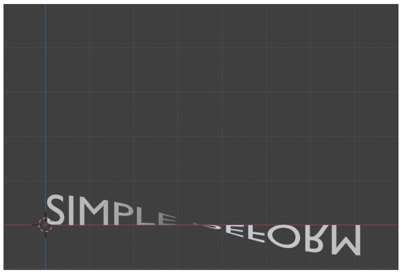
The Simple Deform modifier.

The deformation is calculated in the *local* coordinate space. Be aware that the local axes of an object can differ from the global ones. In the figure above, the global Z axis points up and the local Z axis points at 45°. The deformation is applied along a Deform axis, which can be set by selection from a list (X, Y or Z). By using the *Limits* field, the influence of the modifier can be restricted to a subset of the Deform axis. All distances are measured from the origin of the object. The vertices that are furthest away from the origin on the Deform axis represent the upper and lower limits. The origin of the object and the orientation of the local axes can be defined by an external Deform object (most of the time, an empty).

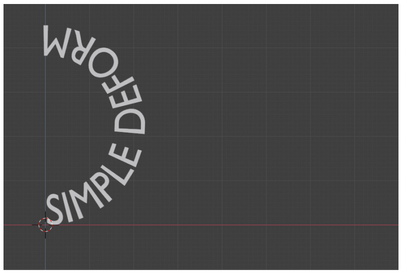
Options

Mode

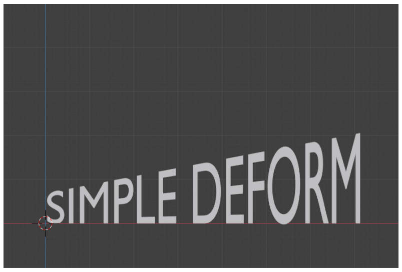
Defines the kind of deformation which will be applied. The figure below shows the four modes, applied to a text object. The origin of the object is at the very left of the text.



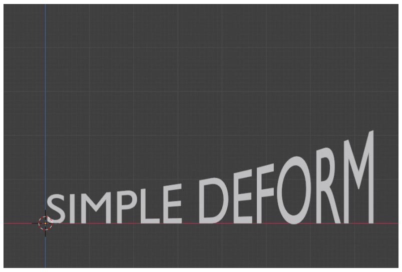
Twist around X axis (180°).



Bend around Z axis (180°).



Taper along X axis (factor = 2).



Stretch along X axis (factor = 0.3).

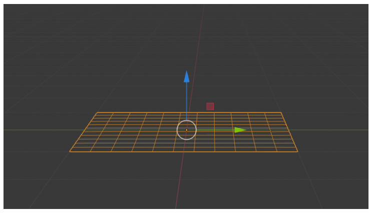
Twist

Rotates the mesh around the specified *Axis*. Each vertex along the Deform axis is rotated around the object's origin. If the origin is inside the object, this results in a twisted appearance. Below the origin, there is a negative rotation and above the origin, the rotation is positive or clockwise. Vertices in the same plane as the origin are not rotated.

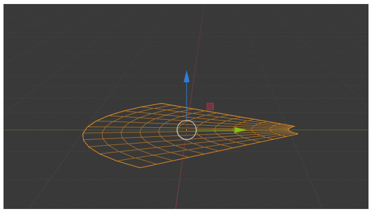
The total amount of rotation is specified by the angle and the rotation at each vertex is thus weighted by the distance of the vertex to the origin of the object. Vertices that are furthest away from the object origin have max rotation, positive or negative.

Bend

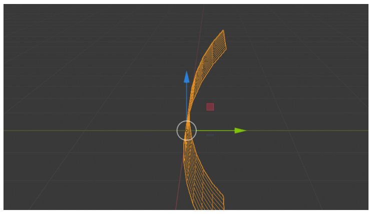
Bends the mesh over the specified *Axis*. The Bend mode is more complex and less intuitive. The picture below shows the same plane but with different Deform and Bending axes.



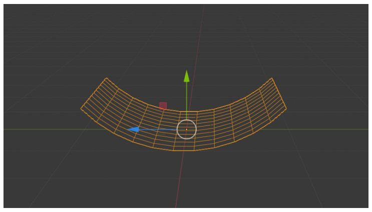
Deform axis X or Y (a).



Deform axis Z (b).



Deform axis X. Local axis Y points down (c).



Deform axis X. Local axis Y points up (d).

Using a mesh plane and setting the Deform axis to X or Y will not result in any deformation (Fig a). You expect that the bending should be something like Figure (c) or (d). In a 3D world however, selecting the X axis introduces an ambiguity because bending along the X axis could result in Figure (c) or (d). The following pairs describe the selected deform axis vs. the desired bending axis: X and Z, Y and Z, Z and X.

In Fig (a), because of the Deform axis X, the Bending is along the Z. All vertices however have the same Z coordinate equal to the local origin. So, no deformation occurs. In Fig (d) the local axes are rotated around the Deform axis X, so that the Bending axis Z points to the left. So, all vertices are bend for their Z coordinate. The further away from the local origin, the more bending.

This explains also the unexpected result of Fig (b). The Deform axis is set here to Z (pointing up). So, according to the pairs above, the Bending axis defaults to X. All vertices of the plane are bent in their X coordinate. The further away, the more rotation occurs. Negative X coordinates are rotated counterclockwise.

Taper

Linearly scales along the specified *Axis*. The scaling factor is weighted by the distance from the origin of the object in the deform axis. No scaling occurs in the plane of the origin of the object. The maximum scaling occurs at the vertices that are furthest away from the local origin. This can be a positive or negative scaling, depending on the location of the origin. If the local origin is within the object, the deformed object appears tapered.

Stretch

Stretches the object along the specified *Axis*. If the local origin is within the object, the deformed mesh looks stretched like pulling a rubber from both sides. With a positive factor, the mesh gets longer in the deformed axis, wider at the borders and thinner at the origin than the original mesh. If the factor is negative, then the mesh is squashed in the deformed axis, thicker at the origin and thinner at the borders.

Angle (Twist & Bend)/Factor (Taper & Stretch)

The total amount of deformation. Can be negative to reverse the deformation.

Axis, Origin

The name of an object that defines the origin and axis of deformation (usually an empty). This object can be:

- Rotated to control the axis (its local *Axis* is now used as the deformation one).
- Moved to control the origin of the deformation.
- Scaled to change the deformation factor.

Restrictions

Limits

You can set the lower and upper limits of the deformation. The upper limit cannot be lower than the lower one. These limits are mapped on the Deform axis.

Lock (Twist, Taper and Stretch modes only)

These controls whether the coordinates along the two other axes are allowed to change or not. E.g. if you *Stretch* your object along its Z axis, it is possible to squash along the X axis only, by locking the Y one.

Vertex Group

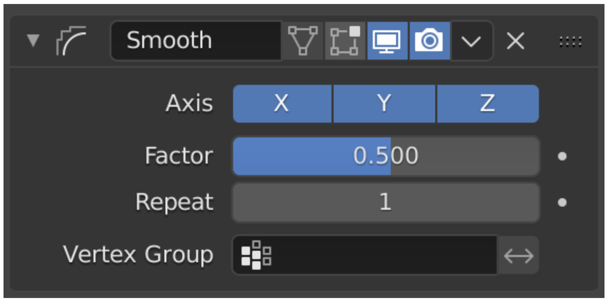
The name of the vertex group that indicates whether and how much each vertex is influenced by the deformation. The amount of influence is determined by the weight in the Weight Paint map.

Smooth Modifier

The *Smooth* modifier smooths a mesh by flattening the angles between adjacent faces in it, just like the Smooth tool in Edit Mode. It smooths without subdividing the mesh, the number of vertices remains the same.

This modifier is not limited to smoothing, though. Its control factor can be configured outside the (0.0 to 1.0) range (including negative values), which can result in interesting deformations.

Options



Smooth

Axis: X Y Z

Factor: 0.500

Repeat: 1

Vertex Group

The Smooth modifier.

Axis

Enable/disable the modifier in the X, Y and/or Z axes directions.

Factor

Controls the smoothing amount. Higher values will increase the effect.

Values outside expected range (above 1.0 or below 0.0) will distort the mesh.

Repeat

The number of smoothing iterations, equivalent to executing the *Smooth* tool multiple times.

Vertex Group

If set, restrict the effect to the only vertices in that vertex group. This allows for selective, real-time smoothing, by painting vertex weights.

Invert <–>

Inverts the influence of the selected vertex group, meaning that the group now represents vertices that will not be deformed by the modifier.

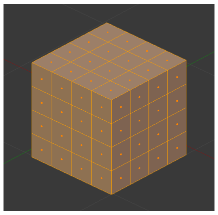
The setting reverses the weight values of the group.

Algorithm

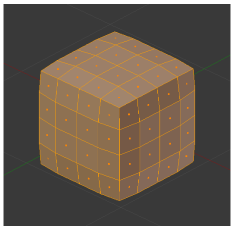
The calculation done by the *Smooth* modifier is a simple and logical one, and can be thought of as the geometric equivalent of blurring images.

Each new vertex position is simply moved towards the average position of all its neighbor vertices (topologically speaking, i.e. the vertices directly connected to it by an edge).

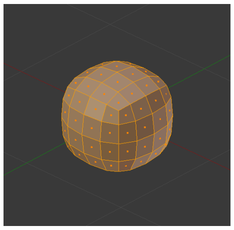
Example



Mesh before smoothing.



Mesh after one smoothing iteration.



Mesh after ten smoothing iterations.

Smooth Corrective Modifier

The *Smooth Corrective* modifier is used to reduce highly distorted areas of a mesh by smoothing the deformations.

This is typically useful *after* an Armature modifier, where distortion around joints may be hard to avoid, even with careful weight painting.

To use this modifier effectively, it is important to understand the basics of how it works.

Rest State
: Used as a reference to detect highly distorted areas. The original vertex locations are used by default.

Smoothing
: Many options for this modifier relate to smoothing which is used internally to correct the distorted regions.

Options

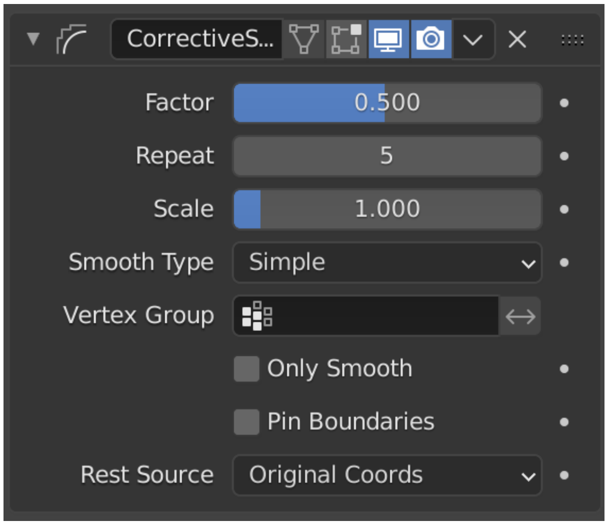
Factor
: The factor to control the smoothing amount. Higher values will increase the effect.

  Values outside expected range (above 1.0 or below 0.0) will distort the mesh.

Repeat
: The number of smoothing iterations, equivalent to executing the *Smooth* tool multiple times.

Scale
: Additional scaling factor to increase the size of the mesh. This is useful because sometimes the Smooth Corrective modifier will introduce

volume loss, especially when used with a rig.



Factor 0.500
Repeat 5
Scale 1.000
Smooth Type Simple
Vertex Group
☐ Only Smooth
☐ Pin Boundaries
Rest Source Original Coords

The Smooth Corrective modifier.

Smooth Type

Select the smoothing method used.

Simple

This simply relaxes vertices to their connected edges.

Length Weight

Uses a method of relaxing that weights by the distance of surrounding vertices. This option can give higher quality smoothing in some cases, by better preserving the shape of the original form.

Vertex Group

If set, restrict the effect to the only vertices in that vertex group. This allows for selective, real-time smoothing, by painting vertex weights.

Only Smooth

This option is included to preview the smoothing used, before correction is applied.

Pin Boundaries

Prevent boundary vertices from smoothing.

Rest Source

Select the source for reference vertex positions that defines the undeformed state.

Original Coordinates

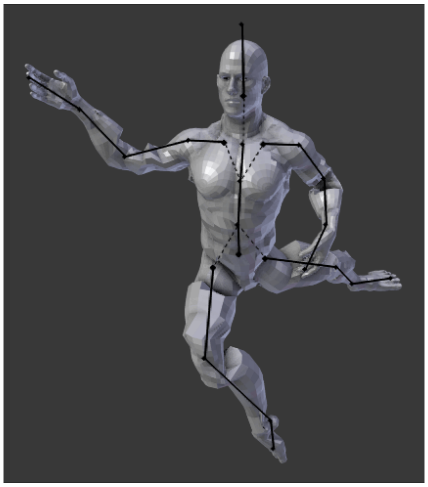
Use the original input vertex positions. This relies on the original mesh having the same number of vertices as the input, modified mesh.

Bind Coordinates

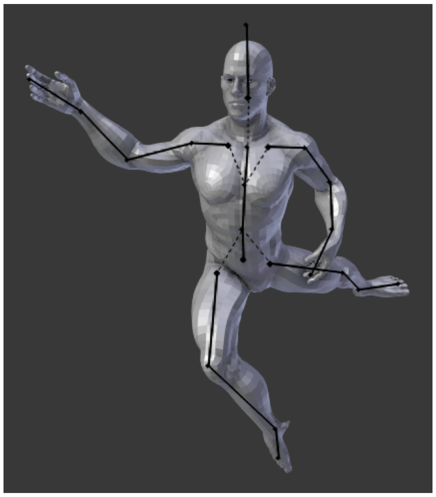
Optionally you may bind the modifier to a specific state. This is required when there are constructive modifiers such as *Subdivision Surface* or *Mirror* in the stack before this modifier.

Example

An example of a rig using bone envelopes and not weight painting.



Armature only.



Armature and Corrective Smooth.

Smooth Laplacian Modifier

The *Smooth Laplacian* modifier allows you to reduce noise on a mesh's surface with minimal changes to its shape.

It can also exaggerate the shape using a negative *Factor*.

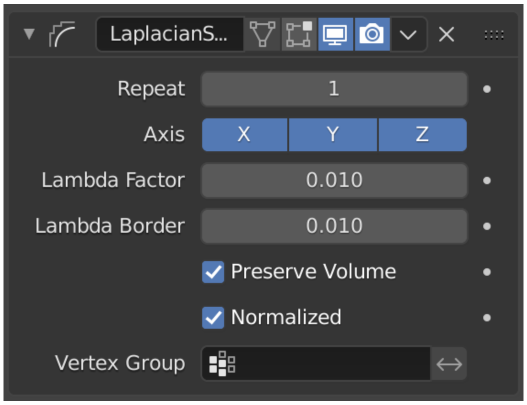
The *Smooth Laplacian* is useful for objects that have been reconstructed from the real world and contain undesirable noise. It removes noise while still preserving desirable geometry as well as the shape of the original model.

This modifier is based on a curvature flow *Laplace Beltrami* operator in a diffusion equation.

Hint

Meshes with a great number of vertices, more than ten thousand (10,000), may take several minutes for processing. You can use small portions of the mesh for testing before executing the modifier on the entire model.

Options



Repeat 1

Axis X Y Z

Lambda Factor 0.010

Lambda Border 0.010

☑ Preserve Volume

☑ Normalized

Vertex Group

The Smooth Laplacian modifier.

Repeat

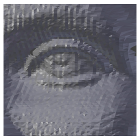
Repetitions allow you to run the smoothing operation multiple times. Each repetition causes the flow curvature of the mesh to be recalculated again, and as a result it removes more noise with every new iteration using a small *Factor* < 1.0.

When on 0, no smoothing is done.

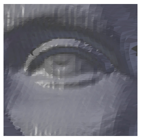
Note

More repetitions will take longer to calculate. So beware of doing so on meshes with a large number of vertices.

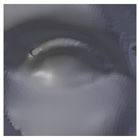
With a factor of 0.5.



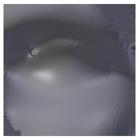
Repeat: 0.



Repeat: 1.

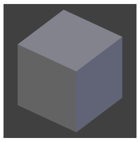


Repeat: 5.

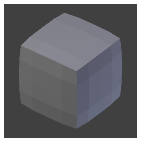


Repeat: 10.

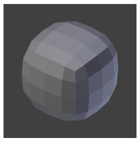
With a factor of 2.0.



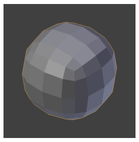
Repeat: 0.



Repeat: 1.

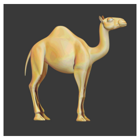


Repeat: 5.

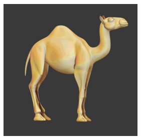


Repeat: 10.

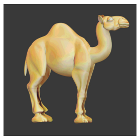
With a factor of -0.5.



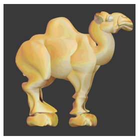
Repeat: 0.



Repeat: 1.



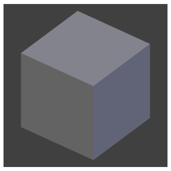
Repeat: 5.



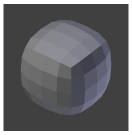
Repeat: 10.

Axis

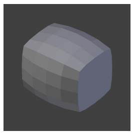
Toggle buttons to enable/disable deforming vertices in the X, Y and/or Z axes directions.



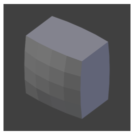
X, Y, Z: Unselected.



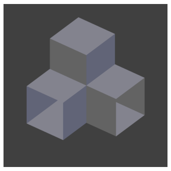
X, Y, Z: Selected.

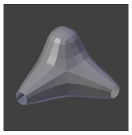


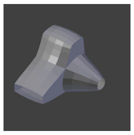
X, Z: Selected.

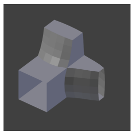


X: Selected.









X, Y, Z: Unselected.

X, Y, Z: Selected.

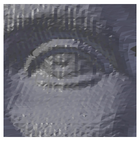
X, Z: Selected.

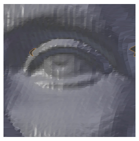
X: Selected.

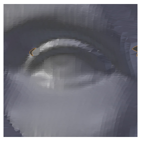
Lambda Factor

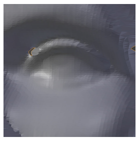
Controls the amount of displacement of every vertex along the flow curvature.

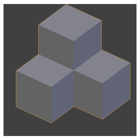
- Using a small *Factor*, you can remove noise from the shape without affecting desirable geometry.
- Using a large *Factor*, you get smoothed versions of the shape at the cost of fine geometry details.
- Using a negative *Factor*, you can enhance the shape, preserving desirable geometry.
- When the *Factor* is negative, multiple iterations can magnify the noise.

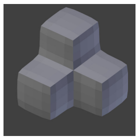

Factor: 0.0.

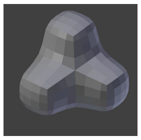

Factor: 0.5.

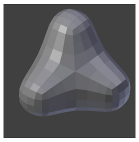

Factor: 2.5.

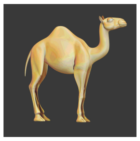

Factor: 5.0.

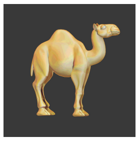

Factor: 0.0.

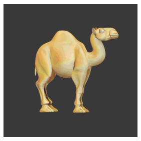

Factor: 1.0.

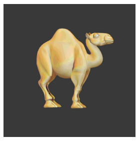

Factor: 10.0.


Factor: 50.0.


Factor: 0.0.


Factor: -20.0.


Factor: -50.0.


Factor: -300.0.

Lambda Border

Since there is no way to calculate the curvature flow on border edges, they must be controlled separately. Border edges are smoothed using a much simpler method, using this property to control the influence.

Positive values will smooth the vertex positions, while negative values will "enhance" them by transforming them in the opposite direction.

With a factor of 2.5.

Border: 0.0.

Border: 1.0.

Border: 2.5.

Border: 10.0.

With a factor of 20.0.

Border: 0.0.

Border: 1.0.

Border: 5.0.

Border: 20.0.

With a factor of -30.0.

Border: 0.0.

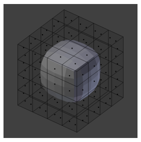
Border: -20.0.

Border: -50.0.

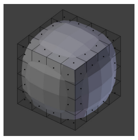
Border: -200.0.

Preserve Volume

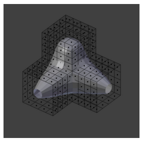
The smoothing process can produce shrinkage. That is significant for large *Factor* or large *Repeat* values. You can use that option to reduce that effect.



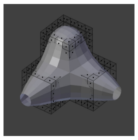
Off.



On.



Off.

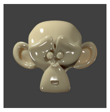


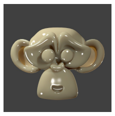
On.

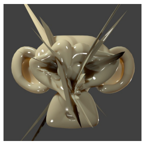
Normalized

When enabled, the results will depend on face sizes. When disabled, geometry spikes may occur.

Original Geometry.

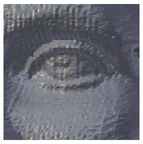
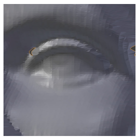
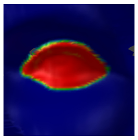
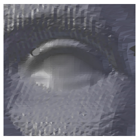
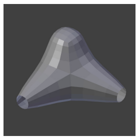
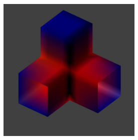
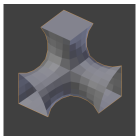
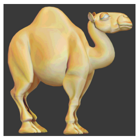
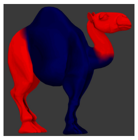
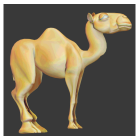

On.


Off.


Off, High Factor.

Vertex Group

A vertex group name, to constrain the effect to a group of vertices only. Allows for selective, real-time smoothing or enhancing, by painting vertex weights.

Original Geometry	**No Group Chosen**	**Vertex Weights**	**Result**
			
			
			

Invert <–>

Inverts the influence of the selected vertex group, meaning that the group now represents vertices that will not be deformed by the modifier.

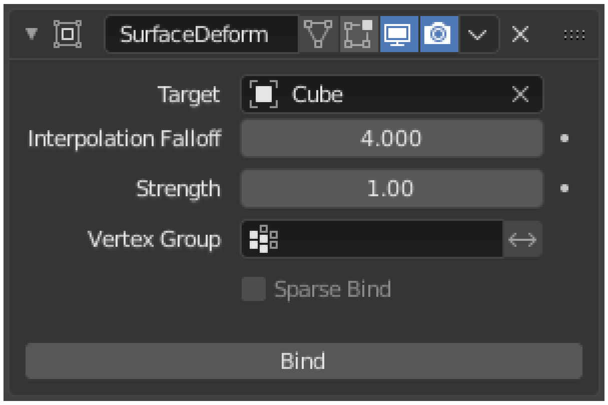
The setting reverses the weight values of the group.

Examples

Surface Deform Modifier

The *Surface Deform* modifier allows an arbitrary mesh surface to control the deformation of another, essentially transferring its motion/deformation. One great use for this is to have a proxy mesh for cloth simulation, which will in turn drive the motion of your final and more detailed mesh, which would otherwise not be suitable for simulation.

Options



Target: Cube
Interpolation Falloff: 4.000
Strength: 1.00
Vertex Group:
☐ Sparse Bind

The Surface Deform modifier.

Target

> The object to which to bind (this setting is unavailable after binding).

> **Warning**
>
> Target Mesh Validity
>
> While there are no restrictions with regard to the modified mesh, the target object's mesh has a few constraints, which if not followed, will prevent a successful binding:
>
> - It must **not** contain edges with more than two faces.
> - It must **not** contain concave faces.
> - It must **not** contain overlapping vertices (doubles).
> - It must **not** contain faces with collinear edges.

Interpolation Falloff

> How much a vertex bound to one face of the target will be affected by the surrounding faces (this setting is unavailable after binding). This essentially controls how smooth the deformations are.

Note

While lower values result in smoother deformations, they may also introduce slight artifacts.

Strength

The overall amount of influence the modifier has on deforming the mesh.

Vertex Group

Allows you to define the influence per vertex.

Invert <–>

Inverts the influence of the selected vertex group, meaning that the group now represents vertices that will not be deformed by the modifier.

The setting reverses the weight values of the group.

Sparse Bind

Only record bind data for vertices that have nonzero weights in the group at the time of bind. This is an optimization, but adding new vertices to the group will require a rebind.

Bind

Bind the current state of the modified mesh to the current state of the target mesh, such that any later change in the target mesh will deform the modified mesh as well. Note that until the bind has been executed, this modifier will have no effect at all.

Unbind

Once the mesh is bound, the *Bind* button changes to *Unbind*. Executing this frees the modified mesh from the target, and resets it to its original shape.

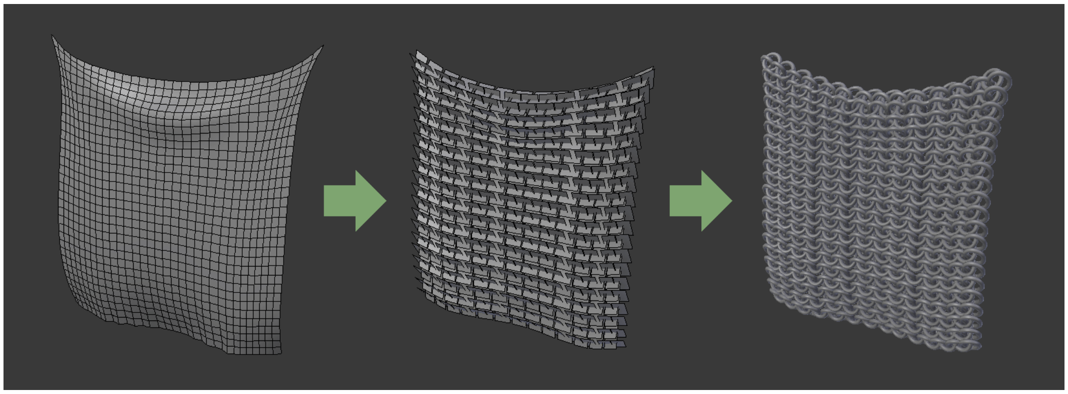
Note

The meshes are bound with regard to global coordinates, but later transformations on the objects are ignored. This means that one can freely transform the target or modified object after binding, without affecting the modified object. The modified mesh will only pick up changes to the target object's mesh itself.

Note

The further a mesh deviates from the target mesh surface, the more likely it is to get undesirable artifacts. This is an inherent characteristic of surface binding in general, so it is recommended to have reasonably well matching meshes, in order to get a good bind.

Example



Cloth simulation copied to an arbitrary mesh with rings as instancing faces.

Volume Displace Modifier

Reference

This modifier is only available for Volume Objects.

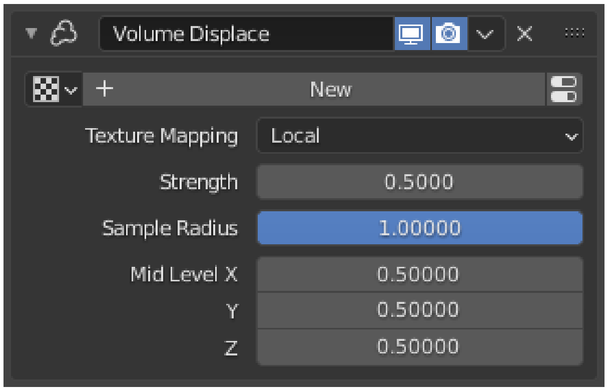
The *Volume Displace* modifier displaces existing volume grids based on a 3D texture. It uses the RGB color channels of the texture to displace the volume into the X, Y and Z direction.

Options

Texture
: The texture that is evaluated at every voxel to determine how far and in what direction to displace.

Note

Grayscale textures lead to stretching along one axis. It's best to use a color texture.



Texture Mapping: Local
Strength: 0.5000
Sample Radius: 1.00000
Mid Level X: 0.50000
Y: 0.50000
Z: 0.50000

The Volume Displace modifier.

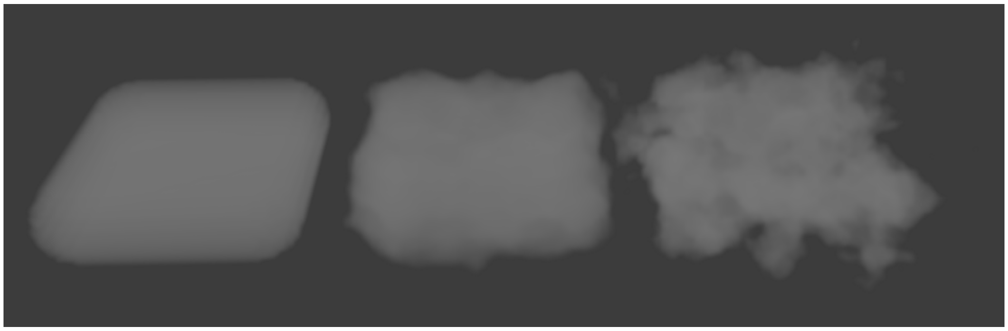
Strength
: Controls how far voxels are displaced.

Sample Radius
: Smaller values result in better performance, but might cut off the volume outside.

Mid Level

This should be modified if the texture offsets the entire volume in one direction and you want to center it again. For performance reasons, the displaced volume should stay close to its original position.

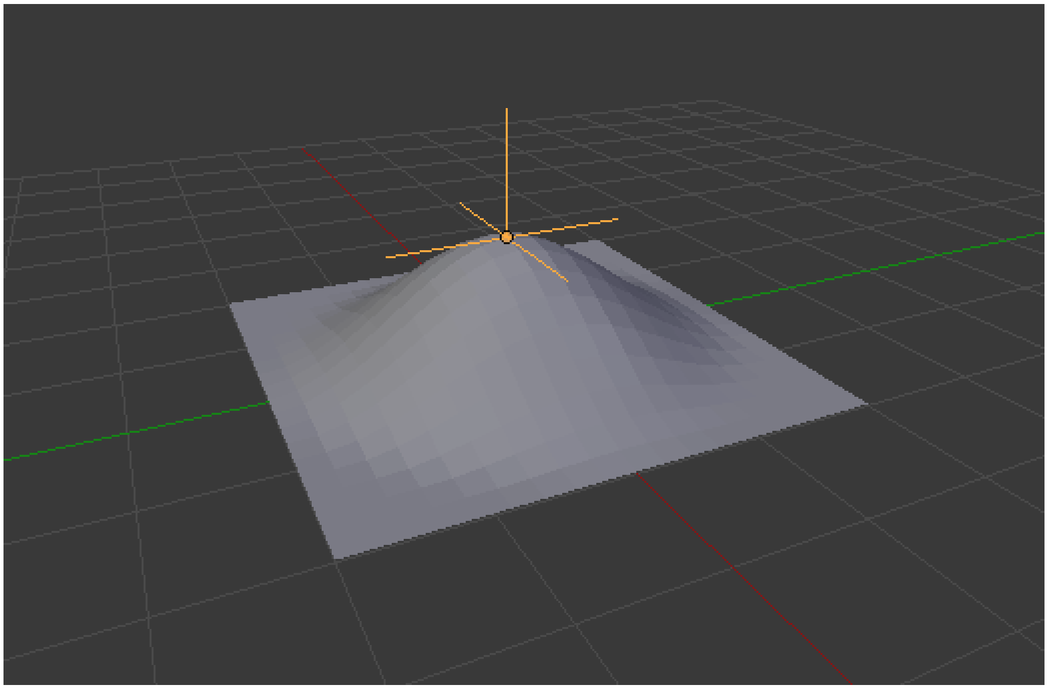
Example



A volume displaced with various strengths.

Warp Modifier

The *Warp* modifier can be used to warp parts of a mesh to a new location in a very flexible way, by using two objects to select the “from” and “to” regions.



A Warp modifier applied to a grid mesh.

This modifier is a bit tricky to understand at first. It requires two points, specified by the two target objects’ origins. The “from” point designates a point in space that is pulled toward the “to” point. It is akin to using the Proportional Editing in Edit Mode.

Options

Object From

The object defining the origin transformation of the warp.

Object To

The object defining the destination transformation of the warp.

Preserve Volume

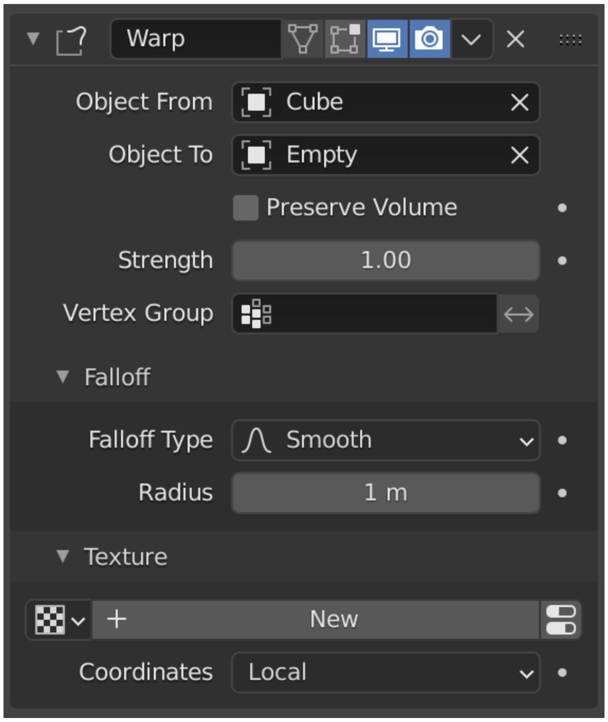
Enables volume preservation when rotating one of the transforms.

Strength

Sets how strong the effect is.

Vertex Group

The name of a vertex group which is used to control the influence of the modifier. If left empty, the modifier affects all vertices equally.



The Warp modifier.

Warp

Object From: Cube

Object To: Empty

☐ Preserve Volume

Strength: 1.00

Vertex Group:

Falloff

Falloff Type: Smooth

Radius: 1 m

Texture

New

Coordinates: Local

Invert <–>

Inverts the influence of the selected vertex group, meaning that the group now represents vertices that will not be deformed by the modifier.

The setting reverses the weight values of the group.

Falloff

Falloff Type

Sets the way the strength of the warp change as it goes from the center of the transform to the *Radius* value. See Proportional Editing for descriptions of the falloff types.

Radius

Sets the distance from the transforms that can be warped by the transform handles.

Texture

Texture

You can finely control which vertices are affected by the warp, and to what extent, using a texture.

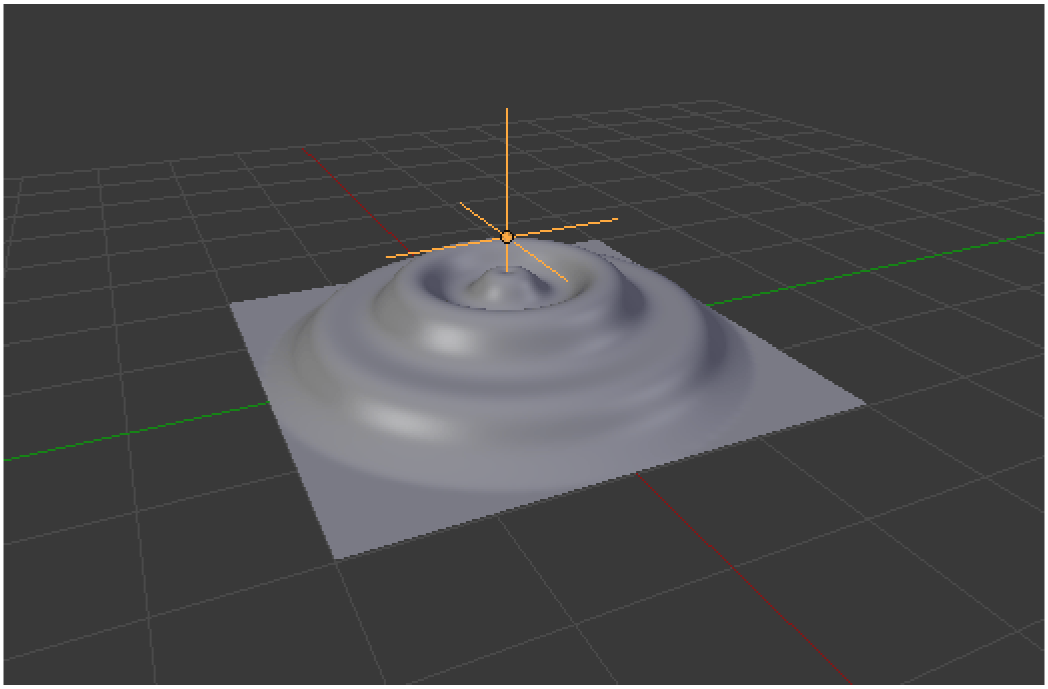
See common masking options for a complete reference.

Usage

The *Warp* modifier can be awkward to use sometimes, and its use case is rather small, but there are a few still. For example, it can be used to have an interactive Proportional Editing that can be used for animations.

Another way to use this modifier is similar to the Deform Modifier. This allows you to deform parts of the mesh without having to make a vertex group.

Examples

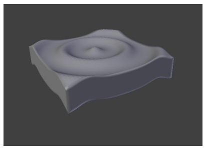


Warp Modifier with a custom falloff curve.

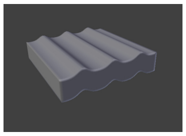
Wave Modifier

The *Wave* modifier adds a ripple-like motion to an object's geometry.

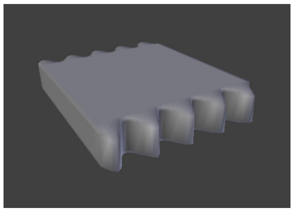
This modifier is available for meshes, lattices, curves, surfaces and texts.



Circular wave front.



Linear wave front.



Motion enabled for X, Normals enabled for Y.

Options

Motion

The wave effect deforms vertices/control points in the Z direction, originating from the given starting point and propagating along the object with circular wave fronts (if both X and Y are enabled), or with rectilinear wave fronts (if only one axis is enabled), then parallel to the axis corresponding to the X or Y button activated.

Cyclic

Repeats the waves cyclically, rather than a single pulse.

Along Normals

For meshes only. Displaces the mesh along the surface normals (instead of the object's Z axis).

X/Y/Z

Restrict displacement along normals to the selected local axes.

Falloff

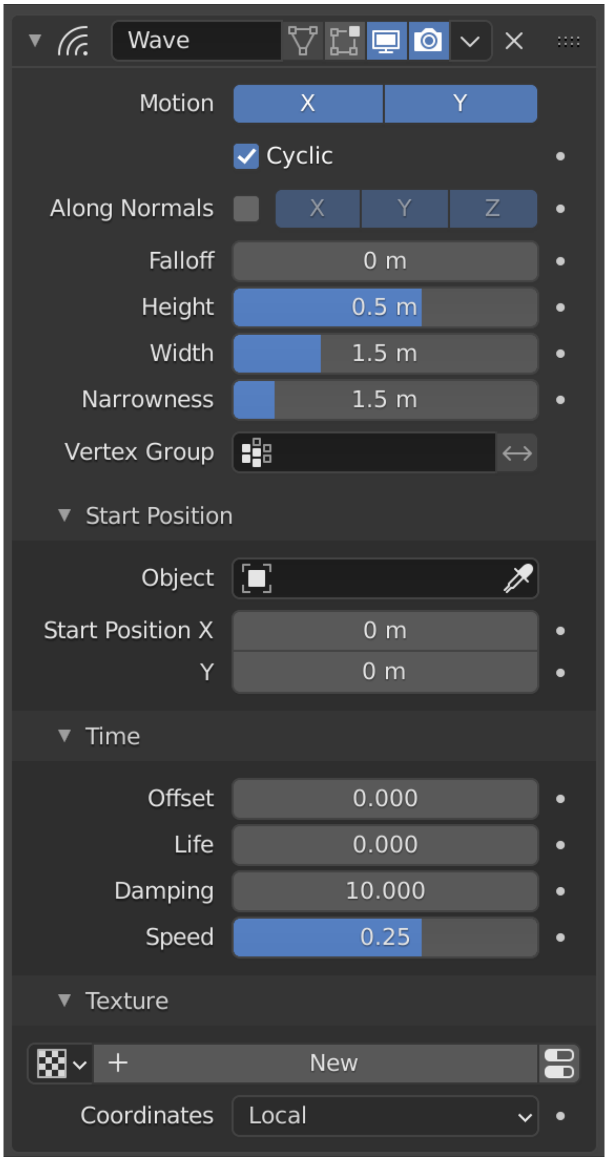
Controls how fast the waves fade out as they travel away from the coordinates above (or those of the *Start Position Object*).

Height

The height or amplitude of the ripple.

Width

Half of the width between the tops of two subsequent ripples (if *Cyclic* is enabled). This has an indirect effect on the ripple amplitude. If the pulses are too near to each other, the wave may not reach the zero Z position, so in this case Blender actually lowers the whole wave so that the minimum is zero and, consequently, the maximum is lower than the expected amplitude. See Technical Details and Hints for more information.


Wave
Motion: X Y
☑ Cyclic
Along Normals ☐ X Y Z
Falloff 0 m
Height 0.5 m
Width 1.5 m
Narrowness 1.5 m
Vertex Group
Start Position
Object
Start Position X 0 m
Y 0 m
Time
Offset 0.000
Life 0.000
Damping 10.000
Speed 0.25
Texture
New
Coordinates Local

The Wave modifier.

Narrowness

The actual width of each pulse: the higher the value the narrower the pulse. The actual width of the area in which the single pulse is apparent is given by `4 / Narrowness`. That is, if *Narrowness* is 1 the *pulse* is 4 units wide, and if *Narrowness* is 4 the *pulse* is 1 unit wide.

Vertex Group

The name of a vertex group which is used to control the influence of the modifier. If left empty, the modifier affects all vertices equally.

Invert <–>

Inverts the influence of the selected vertex group, meaning that the group now represents vertices that will not be deformed by the modifier.

The setting reverses the weight values of the group.

Important

All the values described above are in local object space, i.e. they must be multiplied with the corresponding *Scale* values of the object to get the real dimensions.

Start Position

Object

Use another object as the reference for the starting position of the wave. Note that you then can animate this object's position, to change the wave's origin across time.

Start Position X/Y

Coordinates of the center of the waves, in the object's local space.

Time

Settings to control the animation.

Offset

Time offset in frames. The frame at which the wave begins (if *Speed* is positive), or ends (if *Speed* is negative). Use a negative frame number to prime and pre-start the waves.

Life

Duration of animation in frames. When set to zero, loops the animation forever.

Damping

An additional number of frames in which the wave slowly damps from the *Height* value to zero after *Life* is reached. The dampening occurs for all the ripples and begins in the first frame after the *Life* is over. Ripples disappear over *Damping* frames.

Speed

The speed per frame, of the ripple.

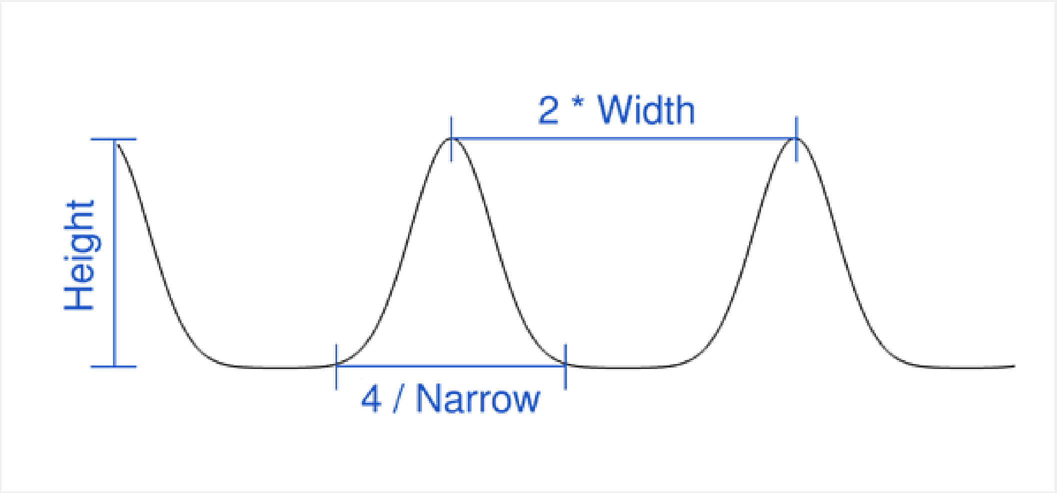
Texture

You can finely control which vertices are affected by the wave, and to what extent, using a texture.

See common masking options for a complete reference.

Technical Details and Hints

The relationship of the above values is described here:



Wave front characteristics.

To obtain a nice wave effect similar to sea waves and close to a sinusoidal wave, make the distance between following ripples and the ripple width equal. That is, the *Narrowness* value must be equal to `2 / Width`. E.g. for *Width* to be 1, set *Narrow* to 2.

Normals

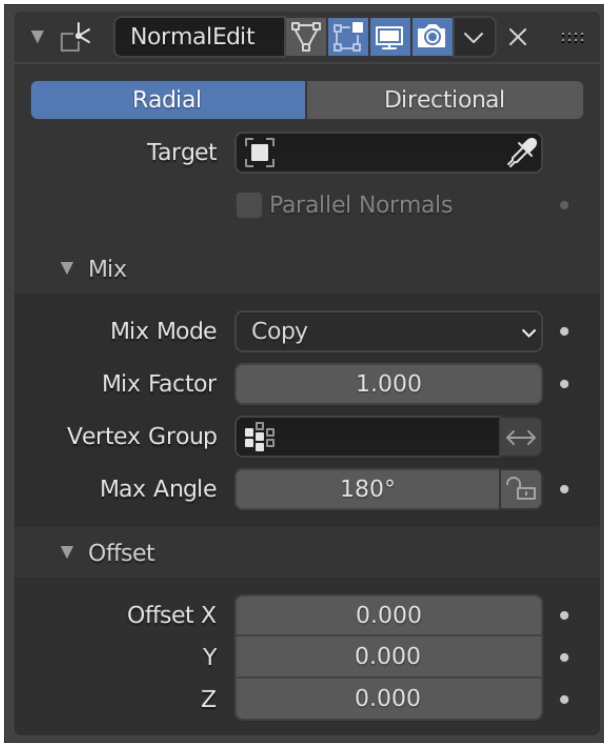
- Normal Edit Modifier
- Weighted Normal Modifier

- Smooth By Angle Modifier

Normal Edit Modifier

The *Normal Edit* modifier affects (or generates) custom normals. It uses a few simple parametric methods to compute them (quite useful in game development and architecture areas), and mixes back those generated normals with existing ones.

Options



Target

Mix Mode: Copy

Mix Factor: 1.000

Vertex Group

Max Angle: 180°

Offset X: 0.000

Y: 0.000

Z: 0.000

Normal Edit Modifier.

Radial

Aligns normals with the `(origin, vertex_coordinates)` vector, in other words all normals seems to radiate from the given center point, as if they were emitted from an ellipsoid surface.

Directional

Makes all normals point (converge) towards a given target object.

Target

Uses this object's origin as reference point when generating normals.

Optional in *Radial* mode, mandatory in *Directional* one.

Parallel Normals

Makes all normals parallel to the line between both objects' origins, instead of converging towards target's origin.

Only relevant in *Directional* mode.

Mix

Mix Mode

How to affect existing normals with newly generated ones.

Note that the *Multiply* option is **not** a cross product, but a faster component-by-component multiplication.

Mix Factor

How much of the generated normals get mixed into existing ones.

Vertex Group

Allows per-item fine control of the mix factor. The vertex group influence can be inverted by using the arrow button to the right.

Max Angle

Forbids new generated normals to have an angle to the original normal above that given threshold. This is useful to prevent extreme changes, that can even lead to inverting the front/back sides of a face, and consequently to shading artifacts.

Lock Polygon Normals (padlock icon)

Prevents flipping (reversing front/back sides) of polygons which normal does not match anymore the side to which point its corners' custom normals. Can also help to avoid shading issues.

Offset

Gives modified object's origin an offset before using it to generate normals.

Only relevant in *Radial* mode if no *Target Object* is set, and in *Directional* mode when *Parallel Normals* is set.

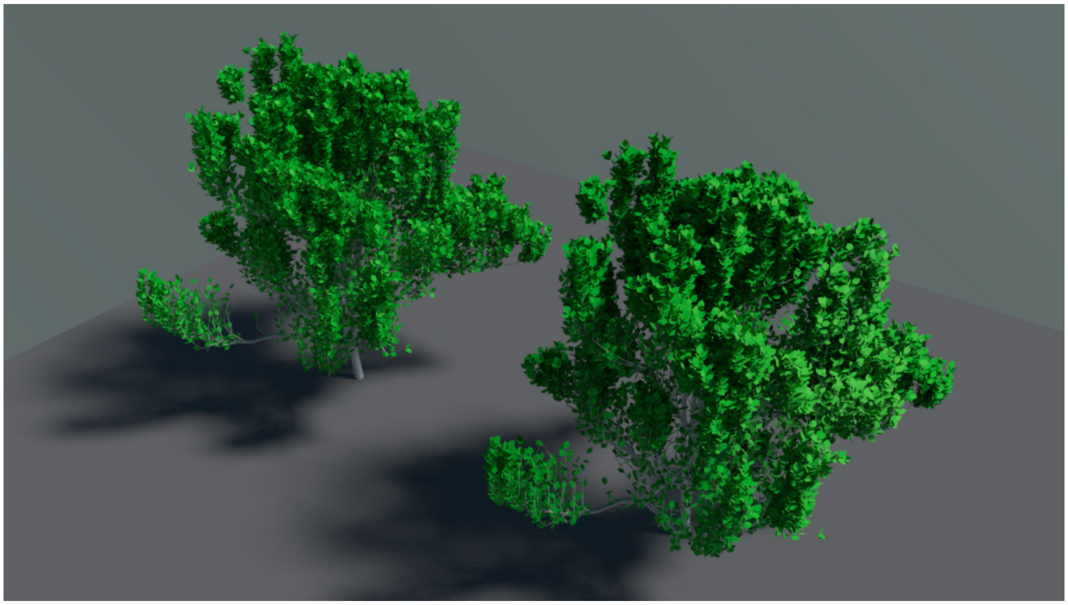
Usage

This modifier can be used to quickly generate radial normals for low-poly tree foliage or “fix” shading of toon-like rendering by partially bending default normals…

Tip

More complex normal manipulations can be achieved by copying normals from one mesh to another, see the Data Transfer Modifier. Some shading effects can also make use of the Weighted Normals modifier.

Example



Editing custom normals to point towards a given direction (blend-file).

The left tree mesh has unmodified normals, while on the right one a *Normal Edit* modifier is used to bend them towards the camera. This shading trick is often used in games to fake scattering in trees and other vegetation.

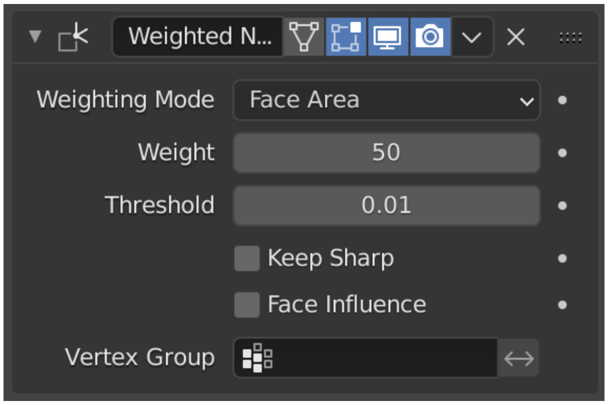
Weighted Normal Modifier

This modifier changes the custom normals of a mesh, using various selectable methods. This can be useful to make some faces appear very flat during shading, among other effects. See Normals for a description of normals and custom normals.

Options

Weighting Mode

The normals around a vertex will be combined to create a custom (per face corner) normal using various weights for each. The *Weighting Mode* defines how to compute the weights.



Weighting Mode: Face Area

Weight: 50

Threshold: 0.01

☐ Keep Sharp

☐ Face Influence

Vertex Group:

Face Area

Weight according to the area of the face that the normal originates. A larger area means that the normal from that face will get a higher weight in final result.

Corner Angle

Weight according to the angle each face forms at the vertex. This is the method Blender uses by default when combining face normals to compute a vertex one.

Face Area and Angle

Weights are obtained by multiplying the face area and corner angle ones.

Weight

Determines how strongly the weights are biased according to the face areas and/or corner angles, a bit like a contrast setting for a picture.

A value of 50 means all faces are weighted uniformly. More than 50 means faces with higher area or angles are given even more weight (more "contrast"). Less than 50 means faces with higher area or angles are given lesser weights (less "contrast").

Threshold

A weight-rounding threshold which means that, if two angles or areas differ by less than that threshold, they will get equal weights.

Keep Sharp

Preserve sharp edges, though smoothing will still happen if there are multiple faces between any two sharp edges.

Face Influence

Use face weights (weak, medium, or strong) as assigned by the Set Strength tool or by the *Set Strength* mode of a Bevel modifier.

For example, if three faces meet at a vertex and have the face weights weak, medium, and strong, then only the normal associated with the strong face will be used to set the final result.

Vertex Group

If a vertex group is specified, the modifier will only affect those vertices. The "arrow" button to its right will invert the selection (only affect the vertices *not* in the vertex group).

Smooth By Angle Modifier

Sets the sharpness of mesh edges based on the angle between the neighboring faces.

Note

This is a geometry nodes asset that is included in the bundled “Essentials” asset library.

Tip

This modifier can easily be added to an object with Shade Auto Smooth or removed with Shade Smooth or Shade Flat.

Options

Angle
  Maximum angle between face normals that will be considered as smooth.

Ignore Sharpness
  Smooth all edges, even if they have been marked as sharp.

Physics

- Cloth Modifier
- Collision Modifier
- Dynamic Paint Modifier
- Explode Modifier
- Fluid Modifier
- Ocean Modifier
- Particle Instance Modifier
- Particle System Modifier
- Soft Body Modifier

Cloth Modifier

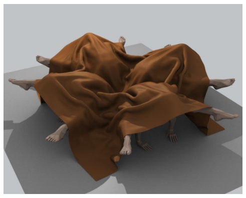
The Cloth modifier is a container for a Cloth Physics simulation. It can be useful for example, to simulate on a low-poly mesh then add a Subdivision Surface Modifier after the Cloth modifier to improve the visual quality of the cloth without drastically increasing simulation times.

Options

As the modifier is only a container its actual options can be configured in the *Physics Properties* tab. See the Cloth Physics Properties for more information.

Example

Cloth example.



Cloth on carved wooden men (made by motorsep).

Cloth example.

Collision Modifier

The Collision modifier is a container for a Collision Physics. Collision physics provide interaction between different physics simulations.

Options

As the modifier is only a container its actual options can be configured in the *Physics Properties* tab. See the Collision Physics Properties for more information.

Example

Deflected particles.

Here is a *Meta* object, using Instancing Vertices to a particle system emitting downwards, and deflected by a mesh cube.

Dynamic Paint Modifier

The Dynamic Paint modifier is a container for a Dynamic Paint Physics simulation.

Options

As the modifier is only a container its actual options are configured in the *Physics Properties* tab. See the Dynamic Paint Physics Properties for more information.

Explode Modifier

The *Explode* modifier is used to alter the mesh geometry by moving/rotating its faces in a way that roughly tracks particles emitted by that object, making it look as if the mesh is being exploded (broken apart and pushed outward).

For this modifier to have any visible effect, there needs to be a particle system on its object. That particle system will control how the mesh is exploded.

Both the number of emitted particles and number of faces determine how granular the *Explode* modifier is. More of each faces and particles will mean more individual pieces.

Here is a demo video showing a cube with a particle system and *Explode* modifier. (blend-file).

Note

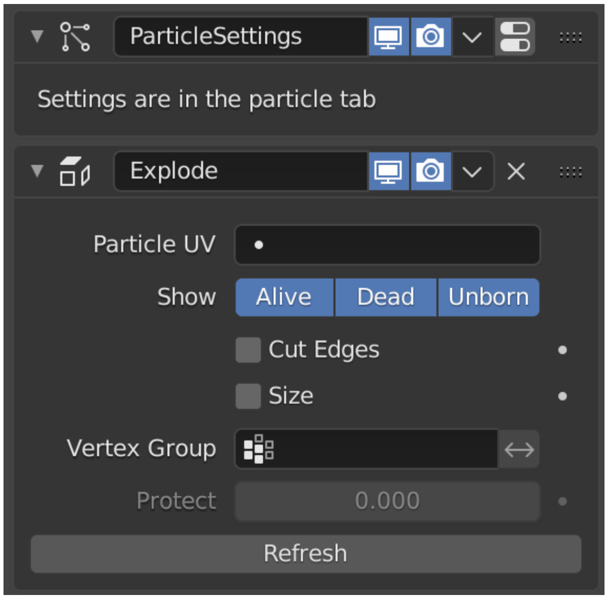
The *Explode* modifier must come after the *Particle System* one in the modifier stack, in order for the former to get required data from the later.

Options

Particle UV

> If set, the U value of the coordinates in that UV Map will be overwritten with the age of the particle attached to the matching mesh face (in proportion, from 0 for not yet born particles, to 1 for dead ones).
>
> The V value is set to a constant 0.5 value.

You can for example vary the color of a fragment (face) during the explosion phase, by using a texture with a color gradient along its U axis.


ParticleSettings

Settings are in the particle tab

Explode

Particle UV

Show: Alive | Dead | Unborn

☐ Cut Edges

☐ Size

Vertex Group

Protect 0.000

Refresh

The Explode modifier, with a Particle System above it.

Show

Unborn

Show faces when their attached particles are unborn.

Alive

Show faces when their attached particles are alive.

Dead

Show faces when their attached particles are dead.

Cut Edges

Split the mesh in pieces based on location of emitted particles, instead of using existing faces. This will typically results in a splitting that appears more random.

Size

Scale each face using the size of its attached particle, once that particle is alive.

Vertex Group

Vertices in this group may not be affected by the *Explode* modifier. Vertices with full weight are not affected at all, while vertices with less weight have a higher chance of being affected.

Vertices with null weight will be treated like those which do not belong to the group at all, and explode normally.

Invert <->

Inverts the influence of the selected vertex group, meaning that the group now represents vertices that will not be deformed by the modifier.

The setting reverses the weight values of the group.

Protect

Clean vertex group edges. Depending on the weights assigned to that vertex group, either completely protect those faces from being affected by the *Explode* modifier (which would happen if the faces had a weight value of 1), or completely remove protection from those faces (which would happen if the faces had a weight value of 0).

Refresh

Refresh data in the *Explode* modifier.

Known Limitations

Dynamic Vertex Weights

This modifier uses the initial vertex weights. Modifiers that dynamically change weights will not influence the explosion as these values are only used once.

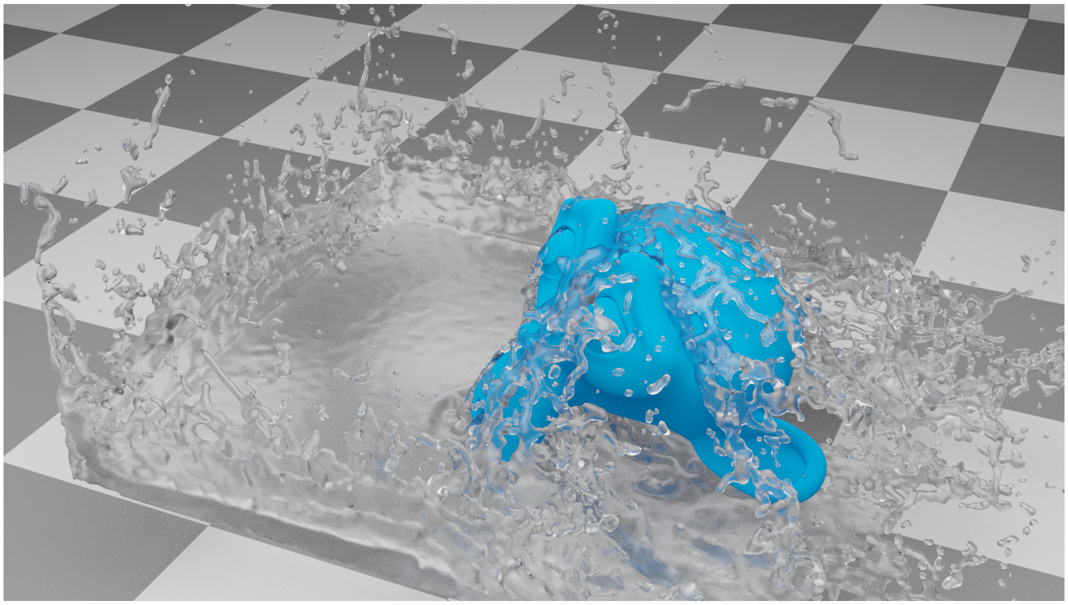
Fluid Modifier

The Fluid modifier is a container for a Fluid Physics simulation. It can be useful for example, to simulate on a low-poly mesh then add a Subdivision Surface Modifier after the Fluid modifier to improve the visual quality of the fluid without drastically increasing simulation times.

Options

As the modifier is only a container its actual options can be configured in the *Physics Properties* tab. See the Fluid Physics Properties for more information.

Example



Example of a liquid simulation.

Ocean Modifier

The *Ocean* modifier is a tool to simulate and generate a deforming ocean surface, and associated texture, used to render the simulation data. It is intended to simulate deep ocean waves and foam.

It is a port from the open source Houdini Ocean Toolkit.

Options

Geometry

  Generate

    Creates a tiled mesh grid that exactly corresponds with the resolution of the simulation data.

    When generating a mesh surface, the existing mesh object is completely overridden with the ocean grid; this also includes any data generated from previous modifiers in the stack. A UV channel is also added, mapping the (0.0 to 1.0) UV space to the simulation grid.

    Repeat X, Y

      Controls the number of times the grid is tiled in X and Y directions. UVs for these tiled mesh areas continue outside of the (0.0 to 1.0) UV space.

  Displace

    Uses the existing geometry rather than replacing it. Vertices are displaced along the local Z axis.

Resolution Viewport, Render

  The main control of quality vs speed in the simulation engine. This determines the resolution of the internal 2D grids generated by the simulation for the 3D Viewport or the final render.

The internal grids are powers of two of the resolution value, so a resolution value of `16`, will create simulation data of size `256×256`. The higher the resolution, the more details will be produced, but the slower it will be to calculate.

Note

When using the *Generate* modifier geometry option, this resolution value also determines the resolution of the generated mesh surface, equal to the resolution of the internal simulation data.

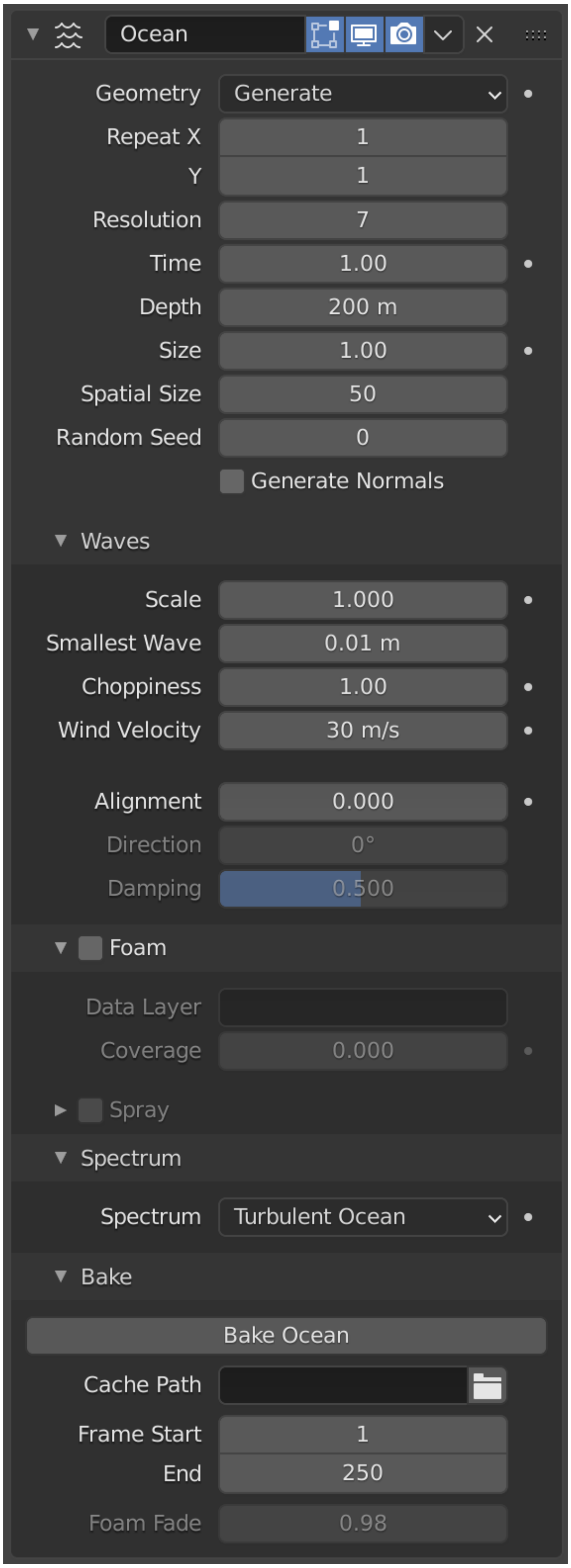
Time

The time at which the ocean surface is being evaluated. To create an animated ocean, you will need to animate this value. The speed that the time value is changing will determine the speed of the wave animation.

Depth

The constant depth of the ocean floor under the simulated area. Lower values simulate shallower waters by producing higher frequency details and smaller waves.

Size



Geometry: Generate
Repeat X: 1
Y: 1
Resolution: 7
Time: 1.00
Depth: 200 m
Size: 1.00
Spatial Size: 50
Random Seed: 0
☐ Generate Normals

Waves
Scale: 1.000
Smallest Wave: 0.01 m
Choppiness: 1.00
Wind Velocity: 30 m/s
Alignment: 0.000
Direction: 0°
Damping: 0.500

☐ Foam
Data Layer:
Coverage: 0.000

☐ Spray

Spectrum
Spectrum: Turbulent Ocean

Bake

Cache Path:
Frame Start: 1
End: 250
Foam Fade: 0.98

A simple scaling factor that does not affect the height of the waves or behavior of the simulation.

The Ocean modifier.

Spatial Size

The width of the ocean surface area being simulated, in meters. This also determines the size of the generated mesh, or the displaced area. Of course, you can scale the object with the *Ocean* modifier in Object Mode to tweak the apparent size in your scene.

Random Seed

A different Seed will produce a different simulation result.

Generate Normals

Simulates extra normal map data. This can be used by the Ocean texture, when mapped to Normals, as a bump map, and enables generating normal map image sequences when baking.

Waves

Scale

An overall scale control for the amplitude of the waves. It approximates the height or depth of the waves above or below zero. Rather than just scaling the ocean object in Z, it scales all aspects of the simulation, displacement in X and Y, and corresponding foam and normals too.

Smallest Wave

A minimum limit for the size of generated waves. Acts similarly to a low-pass filter, removing higher frequency wave detail.

Choppiness

The choppiness of the wave peaks. With a choppiness of 0, the ocean surface is only displaced up and down in the Z direction, but with higher choppiness, the waves are also displaced laterally in X and Y, to create sharper wave peaks.

Wind Velocity

Wind speed in meters/second. With a low velocity, waves are restricted to smaller surface waves.

Alignment

The directionality of the wave shapes due to wind. At a value of 0, the wind and waves are randomly, uniformly oriented. With higher *Alignment* values, the wind is blowing in a more constant direction, making the waves appear more compressed and aligned to a single direction.

Direction

When using *Alignment*, the direction in degrees that the waves are aligned to (using local X axis as reference).

Damping

When using *Alignment*, this will define the amount that inter-reflected waves are damped out. This has the effect of making the wave motion more directional (not just the wave shape).

With a *Damping* of 0.0, waves are reflected off each other in every direction, with a *Damping* of 1.0, these inter-reflected waves are damped out, leaving only waves traveling in the direction of the wind.

Foam

Simulates extra foam data.

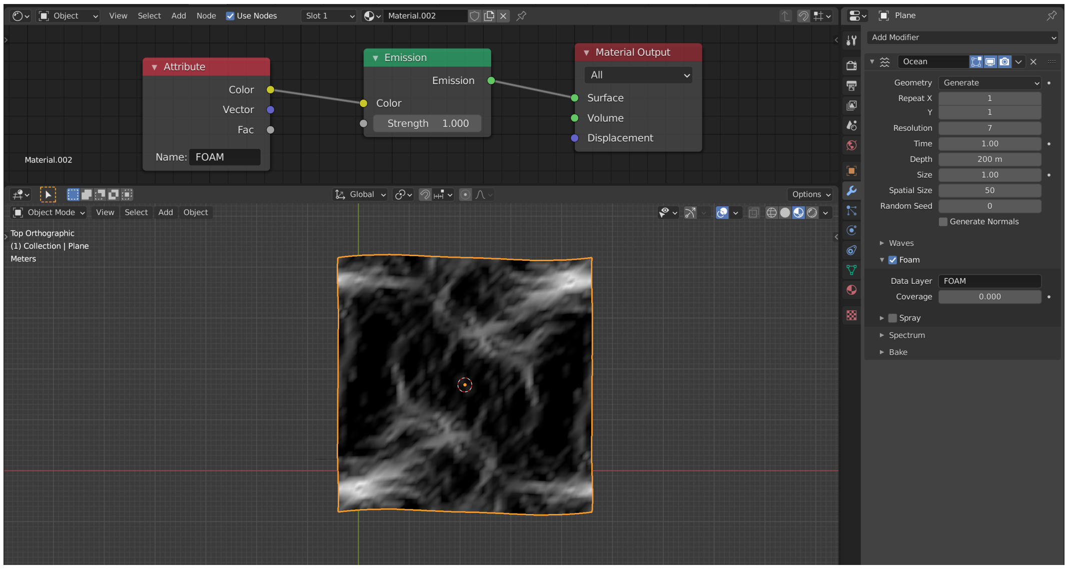
This can be retrieved by the Ocean texture for use in texturing (perhaps as a mask), and enables generating foam map image sequences when baking.

Data Layer

Optional name for the vertex data layer, used by the Ocean Modifier to store foam maps as a Color Attribute. This is required for accessing the foam data in the renderer.

Coverage

Tweaks the amount of foam covering the waves, negative values will reduce the amount of foam (leaving only the topmost peaks), positive values will add to it. Typically ranges from (-1.0 to 1.0).


Object
View
Select
Add
Node
☑ Use Nodes
Slot 1
Material.002
Attribute
Color
Vector
Fac
Emission
Emission
Color
Material Output
Surface
Volume
Displacement
Object Mode
Object
Global
Options
Top Orthographic
(1) Collection | Plane
Meters
Plane
Add Modifier
Ocean
Waves
Spectrum
Bake

Name: FOAM

Strength 1.000

All

Geometry: Generate
Repeat X: 1
Y: 1
Resolution: 7
Time: 1.00
Depth: 200 m
Size: 1.00
Spatial Size: 50
Random Seed: 0
☐ Generate Normals
☑ Foam
Data Layer: FOAM
Coverage: 0.000
☐ Spray

Using foam Color Attributes with a named data layer.

Spray

Generate map of spray direction as a Color Attribute. This map can be used to define the velocities for spray particles.

Spray Map
    Name of the Color Attribute used for the spray direction map.

Invert
    Inverts the spray direction map.

Spectrum

Spectrum
    Used to choose the wave spectrum model to use. Wave spectra are used to describe how energy moves through the waves at different frequencies. Energy travels through waves differently depending on the depth of the water and the wind speed.

Turbulent Ocean

Use for turbulent seas with foam (Phillips).

Established Ocean

Use for a large area, established ocean where the ocean would extend for miles with wind blowing for days allowing the waves to reach a point of equilibrium (Pierson-Moskowitz method).

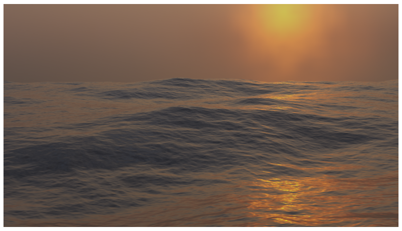
Established Ocean (Sharp Peaks)

Similar to regular *Established Ocean* however, waves will continue to grow with time creating sharper peaks (JONSWAP and Pierson-Moskowitz method). An extra parameter is used to define the sharpness of these peaks.

Shallow Water

Use for shallow water with depths less than about 10 meters which makes it great for small lakes and ponds without heavy wind (JONSWAP and TMA – Texel-Marsen-Arsloe methods).

Examples of different spectra, settings adjusted for each.


Turbulent Ocean.

Established Ocean.

Established Ocean (Sharp Peaks).

Shallow Water.

Sharpness Peak

An artificial factor to control how sharp the peaks of the waves are in the *Established Ocean (Sharp Peaks)* and *Shallow Water* spectrum models.

Fetch

Distance from a lee shore, called the fetch, or the distance over which the wind blows with constant velocity. Used by *Established Ocean (Sharp Peaks)* and *Shallow Water* spectrum models.

Bake

Rather than simulating the ocean data live, it can be baked to files in a given directory. When a simulation is baked, the simulator engine is completely bypassed, and all information for the modifier or texture is retrieved from the baked files.

Baking has the following advantages:

- It is faster to use the stored data rather than recalculating it.
- It allows rendering of ocean data in external renderers.
- It enables more advanced foam maps.

Simulation data is stored as sequences of `OpenEXR` image maps, one for each of displacement, normals, and foam (if enabled to be generated). Upon loading the data from these baked files, when a frame of the bake sequence is read, it is cached in memory. This means that accessing loaded frames subsequent times is fast, not incurring the overhead of drive access.

Since these baked files are plain `OpenEXR`'s, they can also be opened and rendered in any other application or renderer that supports them.

Cache Path

Folder to store the baked EXR files in. The sequences will be in the form `disp_####.exr`, `normal_####.exr`, and `foam_####.exr`, where `####` is the four digit frame number. If the cache path folder does not exist, it will be created.

Frame Start, End

Frames of the simulation to bake (inclusive). The start and end frames of the bake are repeated when accessing frames outside of the baked range.

Foam Fade

Baking also provides improved foam capabilities. When simulating live, the ocean simulator retrieves data for that current frame only. In the case of the foam map, this represents the tips of wave crests for that given frame. In reality, after foam is created by wave interactions, it remains sitting on the top of the wave surface for a while, as it dissipates. With baking, it is possible to approximate that behavior, by accumulating foam from previous frames, leaving it remaining on the surface.

Example

The following example was created and rendered in Blender, notice how the peaks of the waves are white; an effect generated from the foam data.

Particle Instance Modifier

When a Particle Instance modifier is added to an object, the mesh of this object will be duplicated at the location of the particles of the selected particle system from another target object. This means that to use this modifier, you must have at least one other object that has a Particles System on it.

Because of the correlation in which the Particle Instance modifier is influenced by the underlying particle systems on other objects, some of the apparent effects generated by the modifier can look and act vastly different, depending on the underlying settings of the particle systems it is associated with. This is worth taking account of, when it seems that the Particle Instance modifier settings do not return the expected results.

Options

Object
  The target object which has a particle system associated with it.

Particle System
  Which particle system from the target *Object* to apply this modifier to.

Create Instances
  Regular
    When enabled, the modifier will use the regular (parents) particles to duplicate the mesh of the modified object.

  Children
    When enabled, the modifier will use the children particles to duplicate the mesh of the modified object.

  Size

Scale the instanced copies of the mesh by the particle size attribute. When this is disabled, all the copies appear the same size as the origin.

See the particle system's [Render]() and [Children]() panels for particle's size options.

Show

Unborn

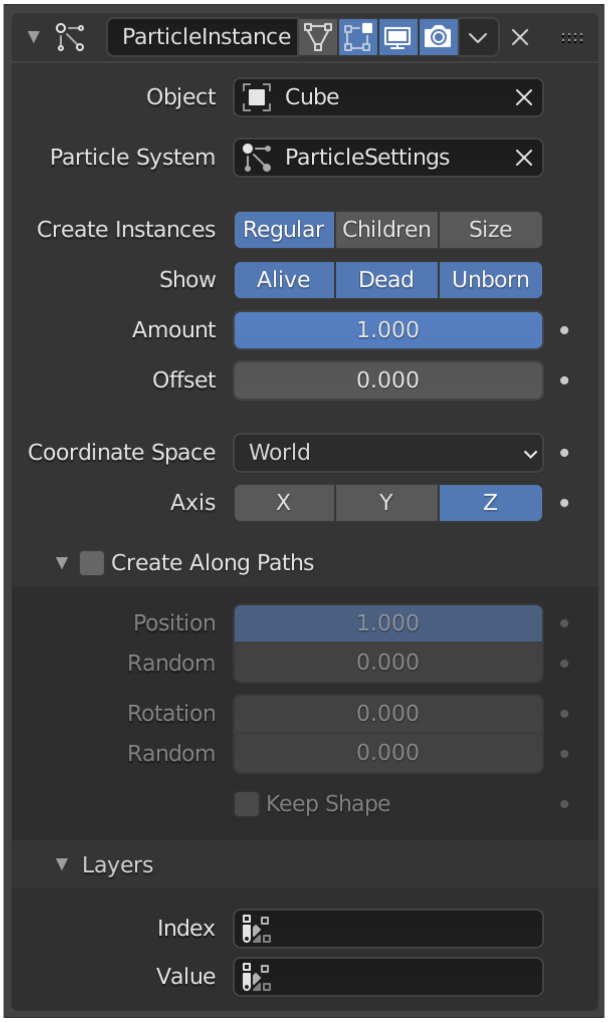
When enabled, the modifier will use the unborn particles to duplicate the mesh of the modified object.

Alive

When enabled, the modifier will use the alive particles to duplicate the mesh of the modified object.

Dead

When enabled, the modifier will use the dead particles to duplicate the mesh of the modified object.



Object: Cube
Particle System: ParticleSettings
Create Instances: Regular | Children | Size
Show: Alive | Dead | Unborn
Amount: 1.000
Offset: 0.000
Coordinate Space: World
Axis: X | Y | Z
☐ Create Along Paths
Position: 1.000
Random: 0.000
Rotation: 0.000
Random: 0.000
☐ Keep Shape
Index:
Value:

The Particle Instance modifier.

Amount

The proportion of particles to be used. Allows you to **randomly** skip particles to adjust the amount of instances.

Warning

The random algorithm used currently only ensures that relative amount to be respected *statistically*. The actual amount of instances generated will differ from the theoretical one, depending on the *Seed* value of the target particle system (and the *Offset* value described below, too).

That deviation is not significant with high number of particles, but it will be highly noticeable with low numbers (e.g. with 100 particles in the target system, and an *Amount* value of `0.1`, it can generate either up to 15 or 5 instances, instead of the 10 expected).

Offset

A relative offset in the range of particles used for instantiation. Allows you to avoid overlapping of the used particles, when the same particle system is used in multiple modifier instances.

Tip

If you want to fully avoid overlaps, your *Offset* value must be at least as high as your *Amount* value.

Coordinate Space

World, Local

Use World Space, or Local Space of the target object (that the particle system is assigned to).

- World space means that the locations of the copies of the modified mesh will depend on the location of the modified object **and** of the target object.
- Local space means that the locations of the copies of the modified mesh will depend only on the location of the modified object.

Axis

Specify which axis of the modified object to use as pole axis to apply the rotation from the instantiated particles.

Create Along Paths

By default, the instances are placed depending on the particles position in the current frame. By enabling *Create Along Paths*, the instance of the modified object follows deforms its shape along the particle path (or the hair strand). This allows you to select the position along the particles path regardless of the current frame.

Tip

You can adjust the particles' path (using the *Path* visualization type) on the Render panel of the *Particle System* tab.

Note

The particle system must be baked, except for *Hair* type or *Keyed* physics.

Position

Specify what percentage of the path that the instance fills, or the position on the path if the *Keep Shape* option is enabled.

Random

Adds randomness to the *Position* value of each instance.

Rotation

Specifies the rotation around the path.

Random

Adds randomness to the *Rotation* value of each instance.

Keep Shape

Enabling this prevents the instance from being deformed, and places it on the path according to the *Position* value.

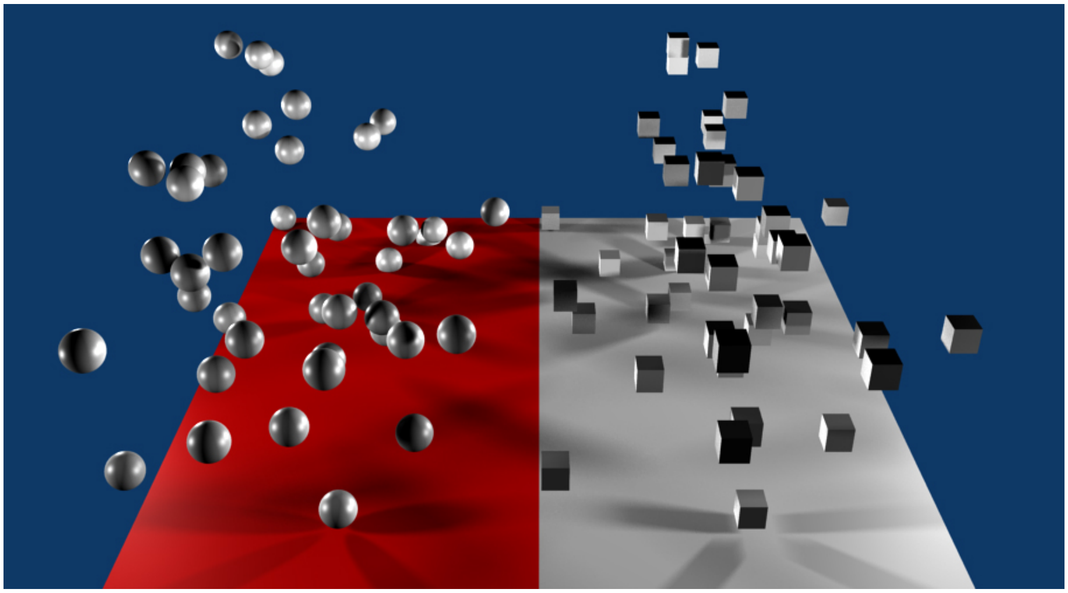
Layers

With these fields you can select the Color Attribute, which will be filled with colors based on the particles information. These Color Attributes can be used, for example, in a shader to add variance to a material.

Index
: A Color Attribute for values based on the particles index.

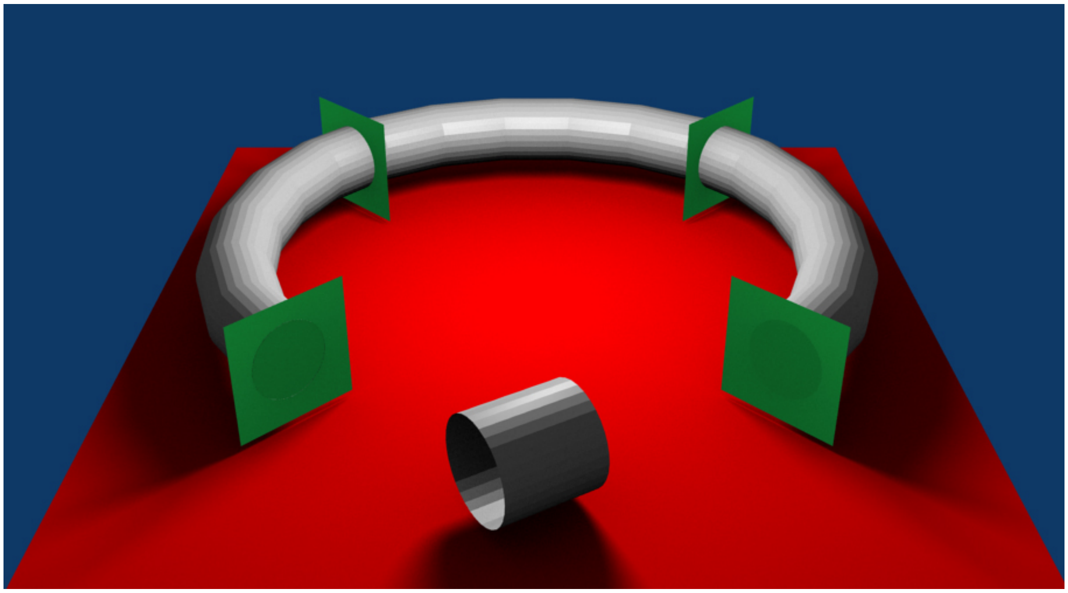
Value
: A Color Attribute for random per-particle values.

Examples



Particle Instance modifier example.

The render above shows a single plane mesh object assigned to two different vertex groups and each of those vertex groups is assigned to a separate and independent particle system, with each particle system being assigned to a different Particle Instance modifier. In the case shown the Particle Instance modifiers are added to a sphere and a cube. See example blend-file.



Create Along Path example.

In this example, a single *Keyed* particle travels through four points (green planes), on an elliptical path. The Particle Instance modifier is added to a cylinder object and then associated with that *Keyed* particle system.

When the *Create Along Paths* is activated, instead of the cylinder location just following the position of the particle, the cylinder mesh is fitted to the shape of the path followed by the particle. The mesh geometry of the object which is deformed can have an impact on how well the deformation is carried out. In the case of the cylinder, it has many loop cuts along its length so that it can bend at those points to deform along the particle path.

The Particle Instance modifier *Create Along Paths* option works for hair (strand) particles as well as with keyed particles. In this case, the mesh of the modifier will follow the length and profile of the hair strands paths.

Note

Strands, when they are generated, instantly die when created, so for the *Create Along Paths* checkbox to be of any use, you must also have the *Dead* checkbox enabled.

Particle System Modifier

The Particle System modifier is a container for Particle Systems.

Note

By default the Particle System modifier does not take into account the modifier stack. Make sure to enable Use Modifier Stack in the Particle properties if you want Particle System modifier to take other modifiers into account.

Options

As the modifier is only a container its actual options are configured in the *Particle Properties* tab. See the Particle Systems Properties for more information.

Converting Particle Systems

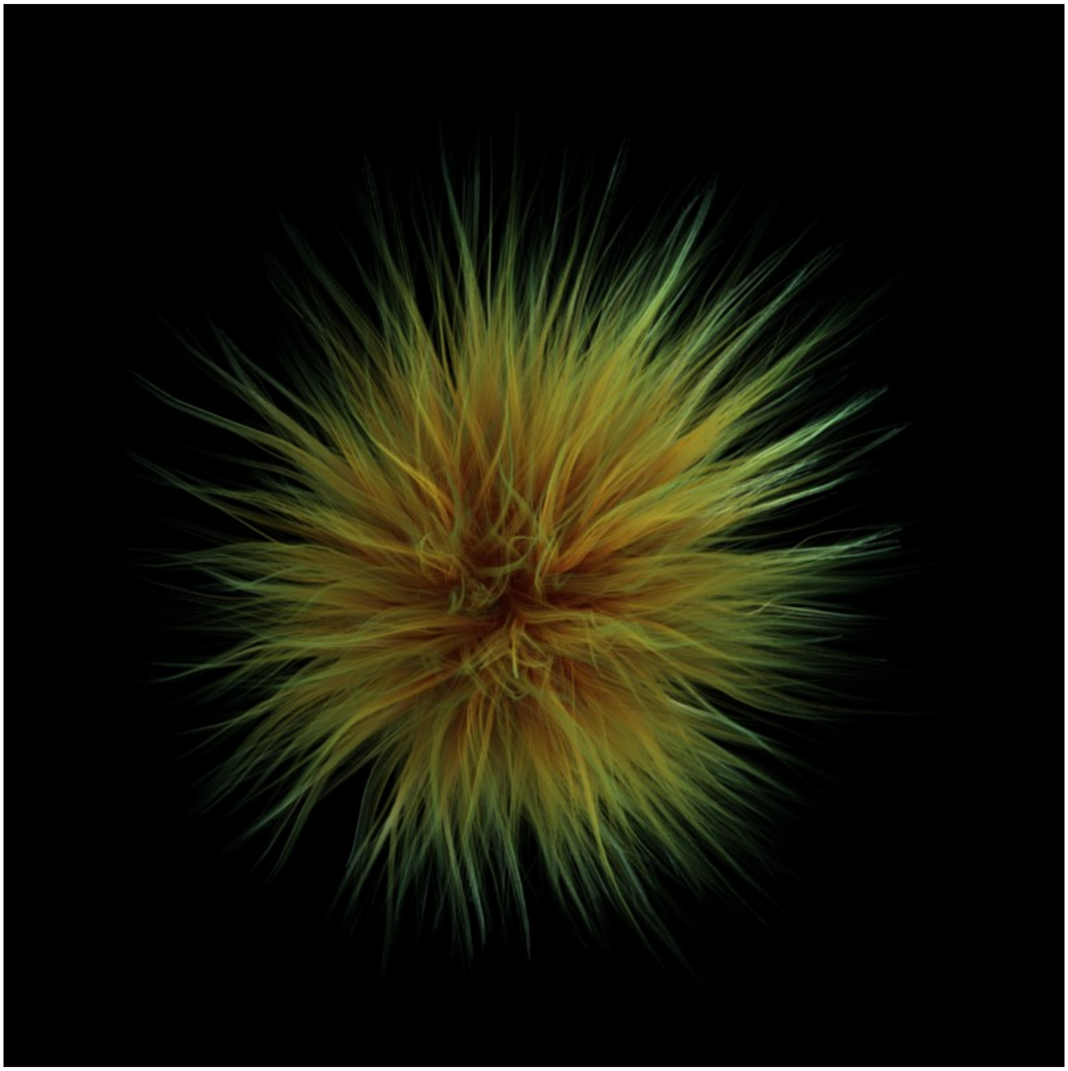
Make Instances Real

Creates a new object of each instanced object or collection. See Make Instances Real for more information.

Convert to Mesh

Converts path particles to mesh objects. See Convert for more information.

Example



Fur made from particles.

Soft Body Modifier

The Soft Body modifier is a container for a Soft Body Physics simulation.

Options

As the modifier is only a container its actual options are configured in the *Physics Properties* tab. See the Soft Body Physics Properties for more information.

Example

The wind cone is a soft body, as the suspension.

Geometry Nodes

- Introduction
- Inspection
- Attributes
- Fields
- Instances
- Baking
- Node-Based Tools
- Gizmos

Node Types

- Attribute Nodes
  - Attribute Statistic Node
  - Domain Size Node
  - Blur Attribute Node
  - Capture Attribute Node
  - Remove Named Attribute Node
  - Store Named Attribute Node
- Input Nodes
  - Constant
    - Boolean Node
    - Color Node
    - Image Node
    - Integer Node
    - Material Node
    - Rotation Node
    - String Node
    - Value Node
    - Vector Node
  - Gizmo
    - Dial Gizmo
    - Linear Gizmo

    - Transform Gizmo
  - Scene
    - 3D Cursor Node
    - Active Camera Node
    - Collection Info Node
    - Image Info Node
    - Is Viewport Node
    - Mouse Position Node
    - Object Info Node
    - Scene Time Node
    - Self Object Node
    - Viewport Transform Node
- Output Nodes
  - Warning Node
  - Viewer Node

- Geometry Nodes
  - Read
    - ID Node
    - Index Node
    - Named Attribute Node
    - Normal Node
    - Position Node
    - Radius Node
    - Selection Node
    - Active Element Node
  - Sample
    - Geometry Proximity Node
    - Index of Nearest
    - Raycast Node
    - Sample Index Node
    - Sample Nearest Node
  - Write
    - Set Geometry Name Node
    - Set ID Node
    - Set Position Node
    - Set Selection Node

  - Operations
    - Bake Node
    - Bounding Box Node
    - Convex Hull Node
    - Delete Geometry Node
    - Duplicate Elements Node
    - Merge by Distance Node
    - Split To Instances Node
    - Sort Elements Node
    - Transform Geometry Node
    - Separate Components Node
    - Separate Geometry Node
  - Geometry to Instance Node
  - Join Geometry Node

- Curve Nodes
  - Read
    - Curve Handle Positions Node
    - Curve Length Node
    - Curve Tangent Node
    - Curve Tilt Node
    - Endpoint Selection Node
    - Handle Type Selection Node
    - Is Spline Cyclic Node
    - Spline Length Node
    - Spline Parameter Node
    - Spline Resolution Node
  - Sample
    - Sample Curve Node
  - Write
    - Set Curve Normal Node
    - Set Curve Radius Node
    - Set Curve Tilt Node
    - Set Handle Positions Node
    - Set Handle Type Node
    - Set Spline Cyclic Node
    - Set Spline Resolution Node

        - Set Spline Type Node
    - Operations
        - Curve to Mesh Node
        - Curve to Points Node
        - Curves to Grease Pencil Node
        - Deform Curves on Surface Node
        - Fill Curve Node
        - Fillet Curve Node
        - Grease Pencil to Curves Node
        - Interpolate Curves Node
        - Merge Layers Node
        - Resample Curve Node
        - Reverse Curve Node
        - Subdivide Curve Node
        - Trim Curve Node
    - Primitives
        - Arc Node
        - Bézier Segment Node
        - Curve Circle Node
        - Curve Line Node
        - Curve Spiral Node
        - Quadratic Bézier Node
        - Quadrilateral Node
        - Star Node
    - Topology
        - Curve of Point Node
        - Offset Point in Curve Node
        - Points of Curve Node
- Instances Nodes
    - Instance on Points Node
    - Instances to Points Node
    - Realize Instances Node
    - Rotate Instances Node
    - Scale Instances Node
    - Translate Instances Node
    - Set Instance Transform Node
    - Instance Transform Node

  - Instance Rotation Node
  - Instance Scale Node
- Mesh Nodes
  - Read
    - Edge Angle Node
    - Edge Neighbors Node
    - Edge Vertices Node
    - Edges to Face Groups Node
    - Face Area Node
    - Face Group Boundaries Node
    - Face Neighbors Node
    - Face Set Node
    - Is Face Planar Node
    - Is Edge Smooth Node
    - Is Face Smooth Node
    - Mesh Island Node
    - Shortest Edge Paths Node
    - Vertex Neighbors Node
  - Sample
    - Sample Nearest Surface Node
    - Sample UV Surface Node
  - Write
    - Set Face Set Node
    - Set Shade Smooth Node
  - Operations
    - Dual Mesh Node
    - Edge Paths to Curves Node
    - Edge Paths to Selection Node
    - Extrude Mesh Node
    - Flip Faces Node
    - Mesh Boolean Node
    - Mesh to Curve Node
    - Mesh to Points Node
    - Mesh to Volume Node
    - Scale Elements Node
    - Split Edges Node
    - Subdivide Mesh Node

        - Subdivision Surface Node
        - Triangulate Node
    - Primitives
        - Cone Node
        - Cube Node
        - Cylinder Node
        - Grid Node
        - Icosphere Node
        - Mesh Circle Node
        - Mesh Line Node
        - UV Sphere Node
    - Topology
        - Corners of Edge Node
        - Corners of Face Node
        - Corners of Vertex Node
        - Edges of Corner Node
        - Edges of Vertex Node
        - Face of Corner Node
        - Offset Corner in Face Node
        - Vertex of Corner Node
    - UV
        - Pack UV Islands Node
        - UV Unwrap Node
- Point Nodes
    - Distribute Points in Volume
    - Distribute Points on Faces
    - Points Node
    - Points to Curves Node
    - Points to Vertices Node
    - Points to Volume Node
    - Set Point Radius Node
- Volume Nodes
    - Operations
        - Volume to Mesh Node
    - Primitives
        - Volume Cube Node

- Simulation Zone

- Material Nodes
    - Replace Material Node
    - Material Index Node
    - Material Selection Node
    - Set Material Node
    - Set Material Index Node
- Texture Nodes
    - Brick Texture Node
    - Checker Texture Node
    - Gabor Texture Node
    - Gradient Texture Node
    - Image Texture Node
    - Magic Texture Node
    - Musgrave Texture Node
    - Noise Texture Node
    - Voronoi Texture Node
    - Wave Texture Node
    - White Noise Texture Node
- Utilities Nodes
    - Color
        - Blackbody Node
        - Color Ramp Node
        - Combine Color Node
        - Mix Color Node
        - RGB Curves Node
        - Separate Color Node
    - Text
        - Join Strings Node
        - Replace String Node
        - Slice String Node
        - Special Characters Node
        - String Length Node
        - String to Curves Node
        - Value to String Node

- Vector
    - Vector Curves Node
    - Vector Math Node
    - Vector Rotate Node
    - Combine XYZ Node
    - Mix Vector Node
    - Separate XYZ Node
- Field
    - Accumulate Field Node
    - Evaluate at Index Node
    - Evaluate on Domain Node
- Math
    - Boolean Math Node
    - Clamp Node
    - Compare Node
    - Float Curve
    - Float To Integer Node
    - Hash Value Node
    - Integer Math Node
    - Map Range Node
    - Math Node
    - Mix Node
- Matrix
    - Combine Matrix Node
    - Combine Transform Node
    - Invert Matrix Node
    - Matrix Determinant Node
    - Multiply Matrices Node
    - Project Point Node
    - Separate Matrix Node
    - Separate Transform Node
    - Transform Direction Node
    - Transform Point Node
    - Transpose Matrix Node
- Rotation
    - Align Rotation to Vector Node
    - Axes to Rotation Node

        - Axis Angle to Rotation Node
        - Euler to Rotation Node
        - Invert Rotation Node
        - Rotate Rotation Node
        - Rotate Vector Node
        - Rotation to Euler Node
        - Rotation to Quaternion Node
        - Quaternion to Rotation Node
    - Deprecated
        - Align Euler to Vector Node
        - Rotate Euler Node
    - For Each Geometry Element Zone
    - Index Switch Node
    - Menu Switch Node
    - Random Value Node
    - Repeat Zone
    - Switch Node

- Group

- Hair Nodes
    - Deformation
        - Blend Hair Curves
        - Displace Hair Curves
        - Frizz Hair Curves
        - Hair Curves Noise
        - Roll Hair Curves
        - Rotate Hair Curves
        - Shrinkwrap Hair Curves
        - Smooth Hair Curves
        - Straighten Hair Curves
        - Trim Hair Curves
    - Generation
        - Duplicate Hair Curves
        - Generate Hair Curves
        - Interpolate Hair Curves

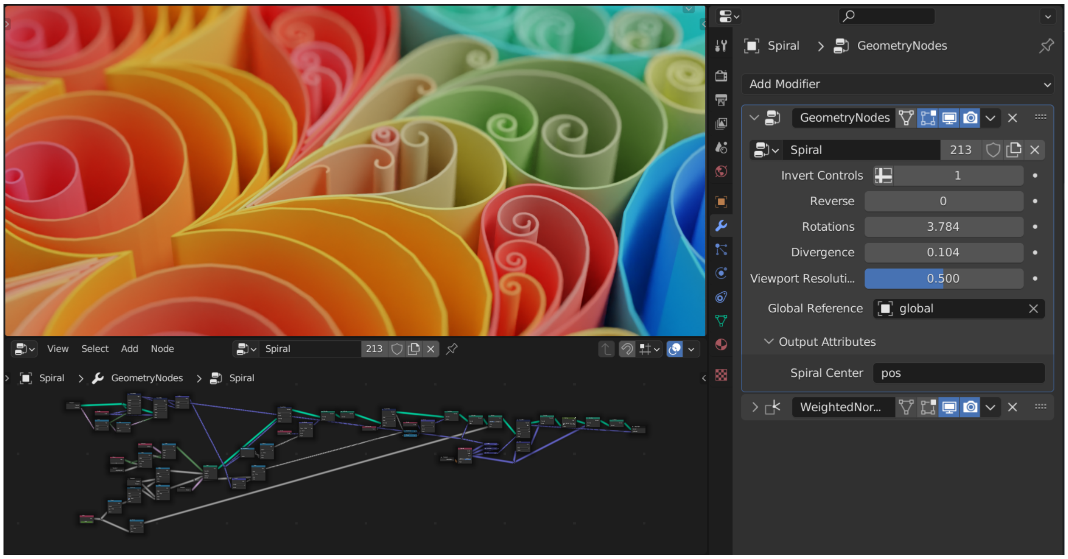
  - Guides
    - Braid Hair Curves
    - Clump Hair Curves
    - Create Guide Index Map
    - Curl Hair Curves
  - Read
    - Curve Info
    - Curve Root
    - Curve Segment
    - Curve Tip
    - Hair Attachment Info
  - Utility
    - Attach Hair Curves to Surface
    - Redistribute Curve Points
    - Restore Curve Segment Length
  - Write
    - Set Hair Curve Profile
- Normals Nodes
  - Smooth By Angle Node Group

Tip

Asset Catalogs that contain geometry node groups will also appear in the add menu.

Introduction

Geometry Nodes is a system for modifying the geometry of an object with node-based operations. It can be accessed by adding a Geometry Nodes Modifier.


View
Select
Add
Node
Spiral
213
Spiral
GeometryNodes
Spiral
Spiral
GeometryNodes
Add Modifier
GeometryNodes
Spiral
213
Output Attributes
WeightedNor...

Invert Controls 1

Reverse 0

Rotations 3.784

Divergence 0.104

Viewport Resoluti... 0.500

Global Reference global

Spiral Center pos

The properties of a Geometry Nodes modifier in the modifier stack.

The geometry node tree connected to a modifier is a Node Group. The geometry from the state before the modifier (the original geometry or the result of the previous modifier) will be passed to the *Group Input* node. Then the node group can operate on the geometry and pass an output to the *Group Output* node, where it will be passed to the next modifier.

Geometry nodes can modify different types of geometry:

- Meshes
- Curves
- Point Clouds
- Volumes
- Instances

The interface of the modifier is described in the Modifier page.

To expand Blender with node-group operators see the description in the Node-Based Tools page.

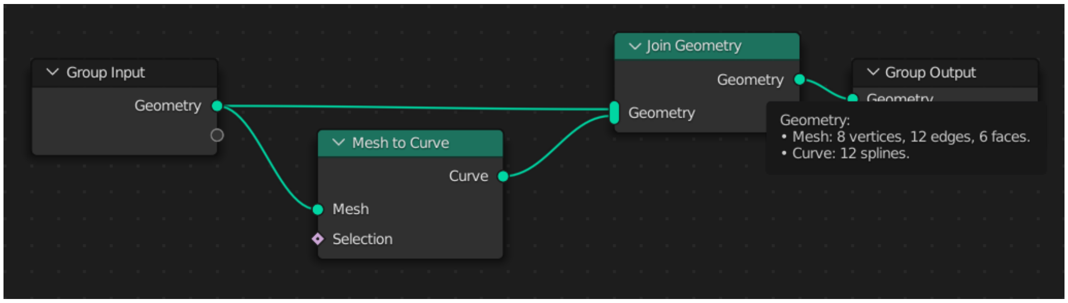
Inspection

Inspecting intermediate values in a geometry node tree is useful while building/understanding one or when trying to figure out why something is not working. Blender provides multiple tools to understand how a node tree is working or why it is not working.

Note

Generally, the inspection tools display data from the last time the node tree has been evaluated. If it has not been evaluated, no information is available.

Socket Inspection



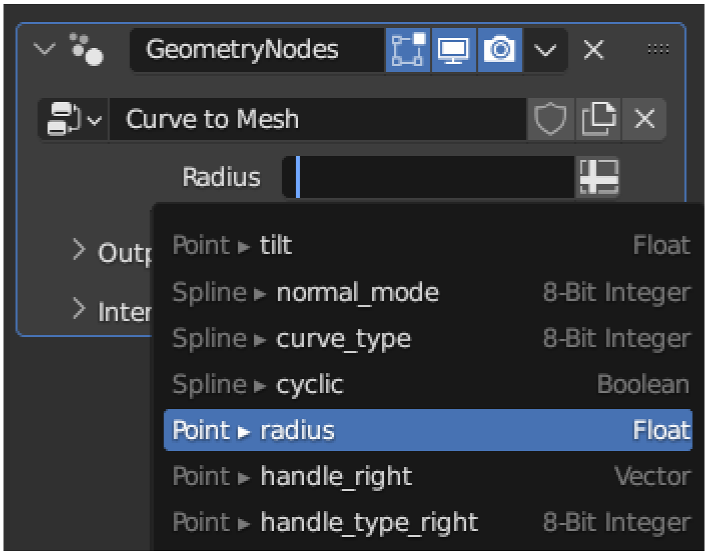
Geometry:
- Mesh: 8 vertices, 12 edges, 6 faces.
- Curve: 12 splines.

Socket Inspection.

Socket inspection shows information about the value in a socket during the last evaluation. For primitive data types such as integers, vectors, and strings the actual value is shown. For geometry sockets only some data about the geometry is stored, including the set of data types the geometry contains, and a count of their elements.

Socket values are only logged from when the node tree was executed, so a node must be connected to the *Group Output* to have a value for inspection. Values are not logged during rendering, to improve performance.

Attribute Search


GeometryNodes
Curve to Mesh
Radius
Outp
Inter
Point ▸ tilt Float
Spline ▸ normal_mode 8-Bit Integer
Spline ▸ curve_type 8-Bit Integer
Spline ▸ cyclic Boolean
Point ▸ radius Float
Point ▸ handle_right Vector
Point ▸ handle_type_right 8-Bit Integer

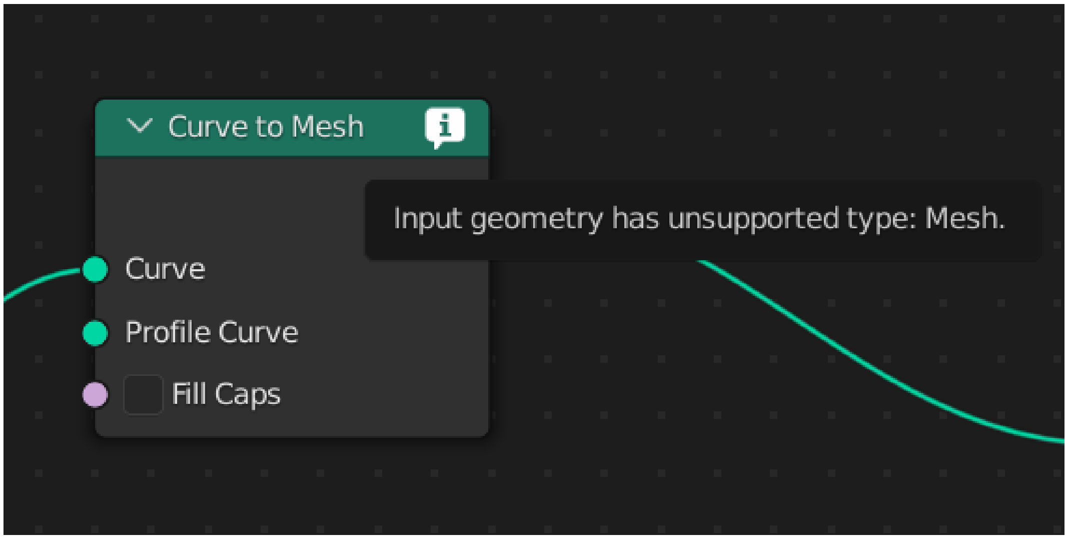
Attribute Search.

The attribute search is shown when clicking on an attribute input in the modifier. It contains a list of all the attributes that were available at that point in the modifier or node execution.

Viewer Node

The Viewer node is used to display intermediate geometry in the Spreadsheet Editor and the Viewport. For more information see Viewer Node.

Node Warnings

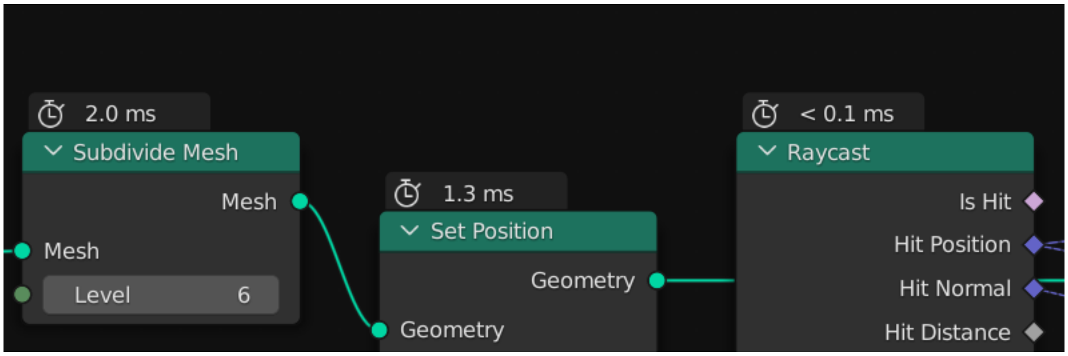

Curve to Mesh
Curve
Profile Curve
Fill Caps

Input geometry has unsupported type: Mesh.

Node Warning.

When the inputs to a node are invalid, it displays a warning in the title. Hovering over the warning icon shows the error message. These warnings are only generated when the node is executed, so a node must be connected to the *Group Output* to have a warning.

Node Timings Overlay


2.0 ms
Subdivide Mesh
Mesh
Mesh
Level 6
1.3 ms
Set Position
Geometry
Geometry
< 0.1 ms
Raycast
Is Hit
Hit Position
Hit Normal
Hit Distance

The node timings overlay.

Node timings show how long a node took to execute the last time the node group was evaluated. They can be turned on in the overlays popover on the top right of the node editor. When a node group is used in multiple places, the timings depend on the context of the node editor, which is displayed in the path on the top left.

Frame nodes display the total time from all of the contained nodes and the *Group Output* node displays the total time for the entire node group.

The displayed timings should only be considered an approximation, since they can also take into account actions like copying or deleting a geometry input that aren't part of the node's operation. Also, when a node uses multiple CPU cores, the evaluation system might work on other nodes at the same time. It's also important to remember that field nodes generally don't do work by themselves, so their execution time is only added to the data-flow nodes they are connected to.

Named Attributes Overlay

The "Named Attributes" overlay allows displaying when a custom named attribute is used by a node or a node group. Named attributes can be used by the Capture Attribute Node, the Named Attribute Node, and the Remove Named Attribute Node, and can be written to, read, or removed.

Using named attributes (as opposed to Anonymous Attributes) can be problematic when the original geometry already has attributes with the specified names. In that case a geometry node group might mistakenly overwrite some essential data. The overlay helps to make detecting that situation easy.

The same data is also available in the Named Attributes panel in the modifier's UI.

Geometry Randomization

Many nodes don't guarantee the order of elements in which they output things. For example, the order of edges coming out of the **Triangulate** node is deterministic but not well defined. The order may change between Blender versions. Therefor, if node setups depend on a specific order, they may break when the Blender implementation changes. Changing the order can often be necessary in order to fix bugs or improve performance.

“Geometry randomization” can be temporarily enabled to see if a blend-file depends on the indices in unstable ways. When enabled, various internal algorithms shuffle the result geometry elements so that any dependence on it would not work anymore. When building setups that are supposed to last a long time, it is recommended to check if they still work with randomization enabled.

To enable it, first enable **Developer Extras** in the preferences. Then search for **Set Geometry Randomization**. The popup allows enabling and disabling the randomization.

Attributes

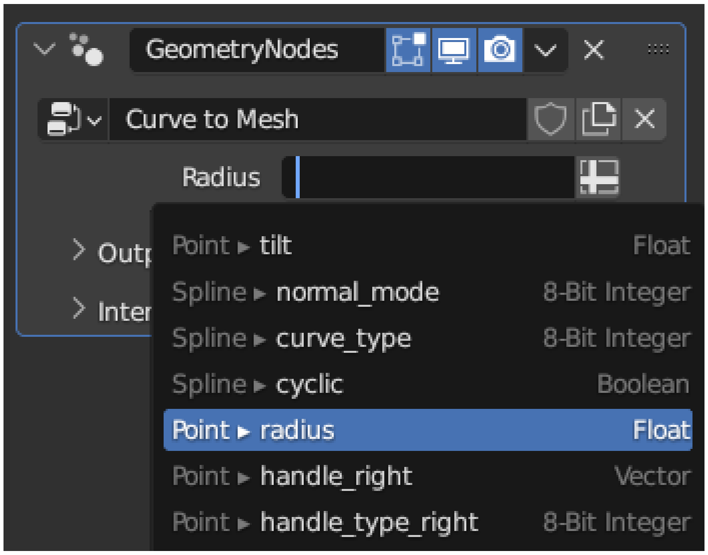
An *attribute* is a generic term to describe data stored per-element in a geometry data-block. For example, every vertex can have an associated number or vector. Attributes can be altered by connecting a value to the *Group Output* node, but also many nodes can change the values of specific attributes.

Note

Attribute data types and domains are converted implicitly where possible, just like node sockets.

Named Attributes

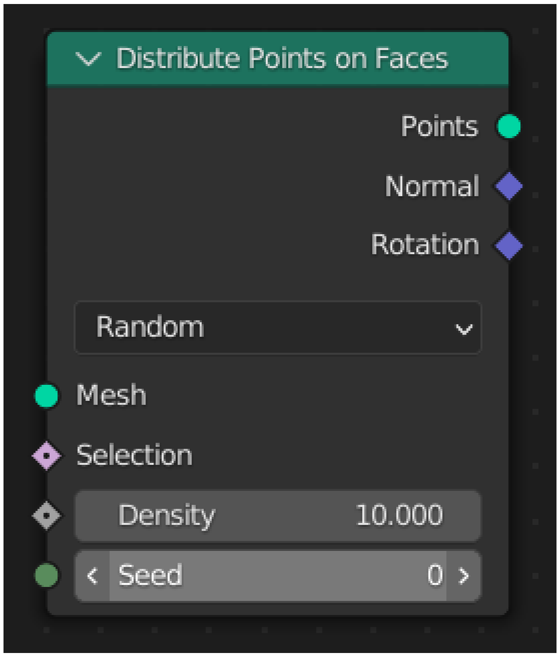
Named attributes are created and used in other areas of Blender like shaders, painting, and UV mapping. In the modifier panel, a named attribute can be used for input by clicking the icon to the right of the value button. The string input input allows you to search and choose existing attributes from the modifier's input geometry.


GeometryNodes
Curve to Mesh
Radius
Outp
Inter
Point ▸ tilt Float
Spline ▸ normal_mode 8-Bit Integer
Spline ▸ curve_type 8-Bit Integer
Spline ▸ cyclic Boolean
Point ▸ radius Float
Point ▸ handle_right Vector
Point ▸ handle_type_right 8-Bit Integer

Attribute Search.

The attribute search gives a bit of context about each attribute. To the left of the menu, the attribute domain is shown followed by the attribute name. To the right of the menu, the attribute data type is shown.

Anonymous Attributes



Random

Density 10.000

Seed 0

The *Normal* and *Rotation* outputs are examples of attribute fields, which refer to an attribute stored on a geometry.

An anonymous attribute is a set of generic data stored on a geometry that doesn't have a name. Usually, attributes exposed in Blender's interface all have names. However, in geometry nodes, attributes can be passed around with node sockets. In these cases, an *Attribute Field* output is created, which is used by nodes to find attribute data in an input geometry.

Anonymous attributes are still stored on the geometry like other attributes, and they are even automatically interpolated when the geometry changes with other nodes, except for a few cases. So generally, if the node link is still accessible, the attribute it references will be available too. However, anonymous attributes cannot be connected to a completely separate geometry that was created from a different source. To transfer attributes between separate geometries, the Sample Index Node or other similar nodes like the Sample Nearest Surface Node can be used.

Attribute Data Types

The type of an attribute is the kind of data stored at each element.

Boolean::	True or false value.
Integer::	32-bit integer.
8-Bit Integer::	Smaller integer with a range from -128 to 127.
Float::	Floating-point value.
Vector::	3D vector with floating-point values.
2D Vector::	2D vector with floating-point values.
Color::	RGBA color with 32-bit floating-point values.
Byte Color::	RGBA color with 8-bit positive integer values.
Quaternion::	Floating point Quaternion rotation.

The above list is in the order of least to most “complex” (An integer can contain more data than a boolean, so it is more complicated). When joining separate geometries together, the more complex data type is preferred when there are matching names. This is particularly important when joining geometry with named attributes with the Join Geometry Node.

To store 2D Vectors (UV maps) and Byte Colors the Store Named Attribute Node must be used because there are no sockets for those types.

Data Conversion

Through the use of Geometry Nodes, data can be converted between types.

Valid conversions:

- Between color and vector – mapping between color channels and vector components.
- Between color and float – the color data is converted to its grayscale equivalent.

- Between float and integer – integers simply become floats, floats are truncated.
- Between float and vector – when a float becomes a vector the value is used for each component. When a vector becomes a float the average of the components is taken.
- Between float and boolean – values greater than 0 are true, true maps to 1, and false maps to 0.

Attribute Domains

The domain of an attribute refers to what type of geometry element the attribute corresponds to. Knowing the domain of an attribute is important because it defines how it may be interpolated and used in nodes and shading. You can use the Spreadsheet Editor to determine the domains of attributes.

- **Point** domain attributes are associated with single locations in space with a position:
    - Vertices of a mesh
    - Points of a point cloud
    - Curve control points
- **Edge** domain attributes are associated with the edges of a mesh.
- **Face** domain attributes are associated with the faces of a mesh.
- **Face Corner** domain attributes are associated with the corners of the faces of the mesh. An example is a UV map attribute.
- **Spline** domain attributes are associated with a group of connected curve control points.
- **Instance** domain attributes exist on the Instances in a geometry. They can be used to store different values on copies of geometry data. Instance domain attributes are only supported in geometry nodes.
- **Layer** domain attributes are associated with a Grease Pencil Layer.

Attributes are automatically interpolated to other domains. For example, when the Position Node is connected to the selection input of the Set Material Node node, the values are interpolated from the *Point* domain to the *Face* domain. Normally, domain conversions use simple averages for values, but *Boolean* data type attributes have special rules for interpolation:

Boolean Domain Interpolation

From	To	Conversion
Point	Edge	An edge is selected if both of its vertices were selected.
Point	Face	A face is selected if all of its vertices were selected too.
Point	Corner	Each corner's value is simply a copy of the value at its vertex.
Point	Spline	A spline is selected if all of its control points were selected.
Edge	Point	A vertex is selected if any connected edge was selected.
Edge	Face	A face is selected if all of its edges are selected.
Edge	Corner	A corner is selected if its two adjacent edges were selected.
Face	Point	A vertex is selected if any of the connected faces were selected.

From	To	Conversion
Face	Edge	An edge is selected if any connected face was selected.
Face	Corner	Each corner’s value is simply a copy of the value at its face.
Corner	Point	A vertex is selected if all connected face corners were selected and it is not a loose vertex.
Corner	Edge	An edge is selected if all corners on adjacent faces were selected.
Corner	Face	A face is selected if all of its corners were selected.
Spline	Point	Each point’s value is simply a copy of the corresponding value of the spline.

Built-In Attributes

Built-in attributes always exist, and cannot be removed. Their data type and domain cannot be changed.

Name	Type	Domain	Notes
`position`	*Vector*	*Point*	Built-in attribute describing vertex or point locations, in the local space of a geometry. Any node that changes the location of points will adjust this attribute, like the Transform Geometry Node and the Set Position Node.
`radius`	*Float*	*Point*	A built-in attribute on point clouds used to set the size for the points in the viewport. Also built-in on curves, where it controls the size of each curve control point when converted to a mesh, or for other operations.
`id`	*Integer*	*Point*	Created by the Distribute Points on Faces to provide stability when the shape of the input mesh changes, and used on instances to create motion blur. Values are expected to be large, with no order. This attribute is used by nodes that generate randomness, like the Random Value Node. Unlike other built-in attributes, this attribute is not required, and can be removed.
`material_index`	*Integer*	*Face*	Used to specify the material slot for every face in a mesh.

Name	**Type**	**Domain**	**Notes**
`sharp_edge`	*Boolean*	*Edge*	Attribute determining if an edge should have flat (rather than smooth) shading enabled in the viewport or a render.
`sharp_face`	*Boolean*	*Face*	Attribute determining if a face should have flat (rather than smooth) shading enabled in the viewport or a render.
`resolution`	*Integer*	*Spline*	Determines the number of evaluated points between two control points of a spline. Only NURBS and Bézier splines have this attribute, for poly splines, the value is always one.
`cyclic`	*Boolean*	*Spline*	Determines whether the spline has a segment that connects its first and last control points.
`handle_left`	*Vector*	*Point*	Describes the location of the left handle of a curve control point, on the side of the curve's start. Only exists when the curve contains a Bézier spline.
`handle_right`	*Vector*	*Point*	Describes the location of the right handle of a curve control point, on

Name	Type	Domain	Notes
			the side of the curve’s end. Only exists when the curve contains a Bézier spline.

Naming Conventions

These attributes do not exist by default, but are used implicitly by certain parts of Blender. The data type of these attributes can be changed, just like any attribute besides the built-in attributes. However, the attributes might be expected by Blender to have a certain type.

Name	Type	Domain	Notes
`velocity`	*Vector*	*Point*	Used to create motion blur when rendering animations.
`rest_position`	*Vector*	*Point*	Holds the position of points or vertices from before a geometry is deformed procedurally. Can be created automatically before Shape Keys and Modifiers are evaluated with the Add Rest Position option.
`surface_uv_coordinate`	*2D Vector*	*Curve*	Used to describe curve attachment locations on a

Name	Type	Domain	Notes
			mesh surface, typically used for the hair system.
`crease_vert`	*Float*	*Point*	Vertex attribute used by the Subdivision Surface modifier. The values are expected to be in a range of 0 and 1.
`crease_edge`	*Float*	*Edge*	Edge attribute used by the Subdivision Surface modifier. The values are expected to be in a range of 0 and 1.
`uv_seam`	*Boolean*	*Edge*	True if an edge is considered a boundary between UV islands when unwrapping.
`bevel_weight_vert`	*Float*	*Point*	Used as vertex control for the bevel modifier.
`bevel_weight_edge`	*Float*	*Edge*	Used as edge control for the bevel modifier.
`sculpt_face_set`	*Integer*	*Face*	Used by the Sculpt Face Sets Feature.

Name	Type	Domain	Notes
`sculpt_mask`	*Float*	*Point*	Used by the Sculpt Masking Feature.
`sculpt_face_set`	*Integer*	*Face*	Used by the Sculpt Face Sets Feature.

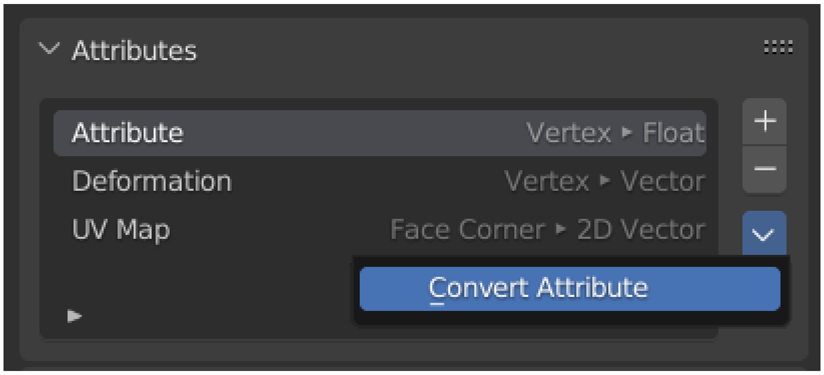
Custom Attributes

Vertex groups, UV maps and Color Attributes are available as attributes in geometry nodes. They are referred to by their name. Naming collisions (e.g. a vertex group and a UV map with the same name) should be avoided. If there is a naming collision, only one of the attributes is accessible in geometry nodes.

Attributes with any other name can also be created by nodes, when the name is used for the first time.

Note that geometry nodes does not always produce e.g. vertex groups if a node like Join Geometry is used. Similarly, if the data type of a vertex group attribute is changed from the initial “Float” type, the attribute will no longer be a vertex group.

Attribute Conversion Operator


Attributes
Attribute
Vertex ▸ Float
Deformation
Vertex ▸ Vector
UV Map
Face Corner ▸ 2D Vector
Convert Attribute

This operator found in the *Attributes* panel of the property editor can change the domain or data type of an attribute.

Due to ongoing development in the area of attributes, many areas of Blender can not yet work with the generic (identified with a name, stored on any domain with any data type) attributes used by geometry nodes. That makes this operator an essential workaround in some cases where existing tools must be used with data generated from geometry nodes.

Mode

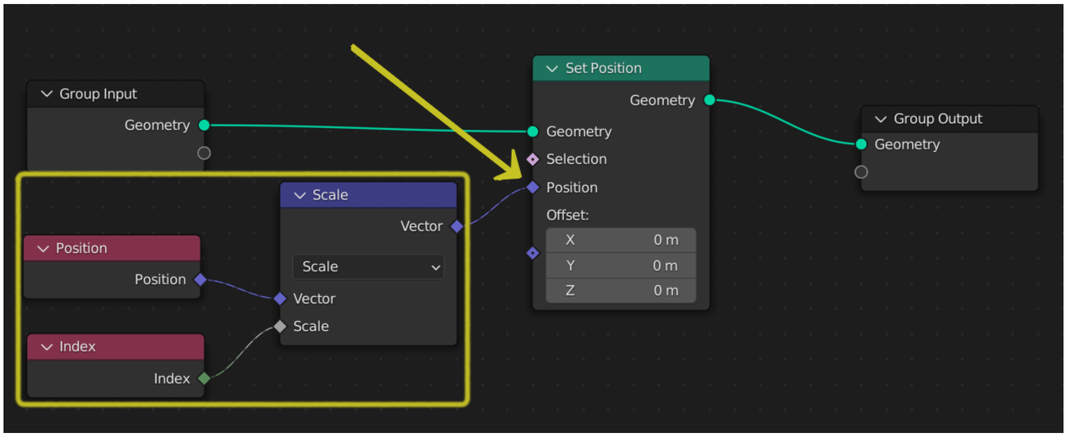
Generic::	Interpolate and convert the attribute between the domains and data types described on this page.
Vertex Group::	Create a Vertex Group from the attribute, which corresponds to a float attribute on the point domain.

Note

This operator only works on *original* object data, not including the results of modifiers, so any attributes added or changed by geometry nodes will not be affected. To change the type of an attribute generated procedurally, modifiers must be applied.

Fields

Fundamentally, a field is a function: a set of instructions that can transform an arbitrary number of inputs into a single output. A field's result can then be calculated many times with different input data. They are used all over geometry nodes to allow calculations that have different results for every element (mesh vertices, faces, etc.).


Group Input
Geometry
Set Position
Geometry
Geometry
Selection
Position
Offset:
X 0 m
Y 0 m
Z 0 m
Group Output
Geometry
Scale
Vector
Scale
Vector
Scale
Position
Position
Index
Index

A field input to a node.

For example, in the figure above, the field connected to the "Set Position" node depends on two inputs, Position and Index, and transforms them into a vector using a single instruction.

Field Visualization

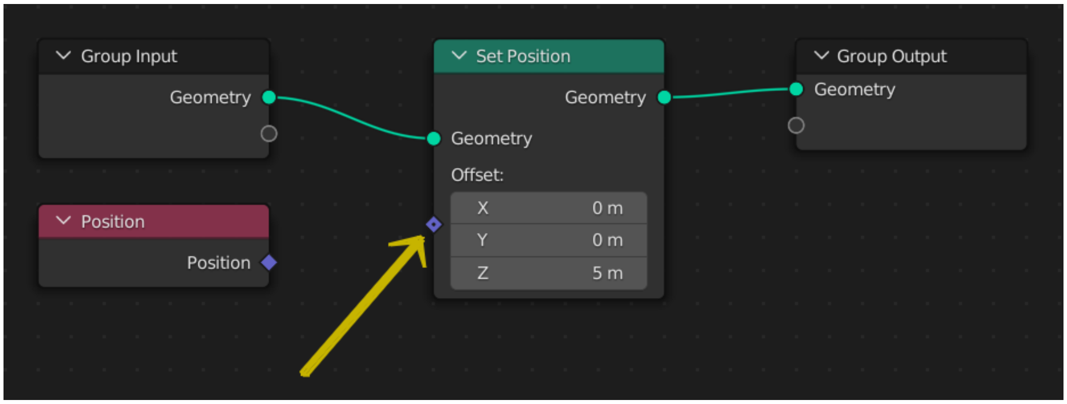
Socket shapes are used to convey which sockets are fields and which are regular data. There are three possible socket shapes, each visualizing its "field status":

Circle::	The socket requires a single real value, it cannot accept a field input. For output sockets, this means the node always outputs a single value.
Diamond::	

The socket can accept a field input, or it outputs a field. A constant single value can be connected to these sockets, but then the output will often not vary per element.

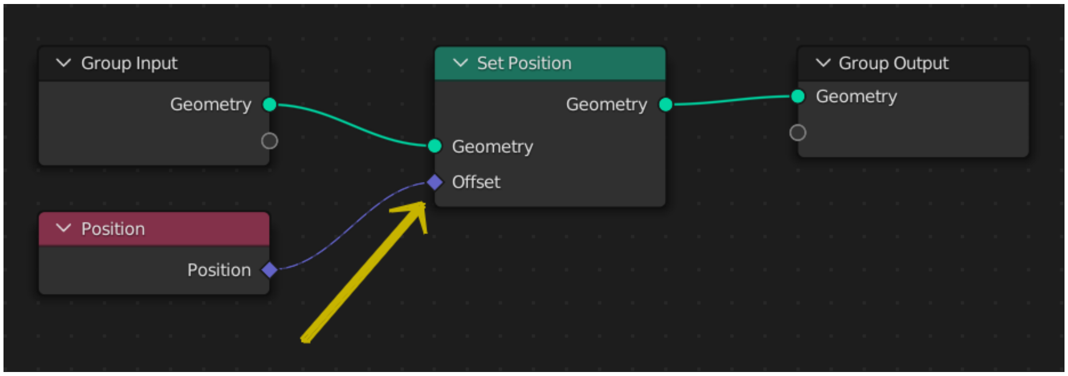
Diamond with Dot::

The socket can be a field, but it is currently a single value. This is helpful because it allows tracking where single values are calculated, instead of a field with many different results. It also means that [Socket Inspection]() will show the value instead of field input names.


Group Input
Geometry
Position
Position
Set Position
Geometry
Geometry
Group Output
Geometry

Offset:
X 0 m
Y 0 m
Z 5 m

The socket shape is a diamond with a dot, meaning the field has the same value for every element. Every point will be moved up by 5 m.


Group Input
Geometry
Position
Position
Set Position
Geometry
Geometry
Offset
Group Output
Geometry

The socket shape is a diamond and the field input now has a varying input. In other words, the value can be different for every element. In this case, the position will be doubled, since the offset for every point is the point’s position.

Tip

Often it is desired to extract a single value from a field. While it doesn’t make sense conceptually to simply change a field into a single value, the Sample Index Node or the Attribute Statistic Node can be used to retrieve a single value from a field evaluated on a geometry.

When a connection is made between two node sockets that support fields the node connection will be drawn as a dashed line. If you make the mistake of connecting a non-field socket to a field socket, the connection will be drawn as a solid red line indicating that there is an error.

Node Types

Nodes can be separated into two categories: data flow nodes that usually pass geometry, and field nodes that operate on data per-element. Field nodes can be input nodes that bring geometry data into the node tree, or function nodes that operate on that data.

Data Flow Nodes

Nodes with a geometry input and a geometry output will almost always be data flow nodes. Meaning that they actually change geometry data that will be outputted from the Geometry Nodes modifier.

Function Nodes

Nodes with diamond socket inputs and outputs are field nodes, and resemble the instructions that will be evaluated by data flow nodes. Examples of function nodes are the math nodes and also more complex nodes like the Geometry Proximity Node.

Input Nodes

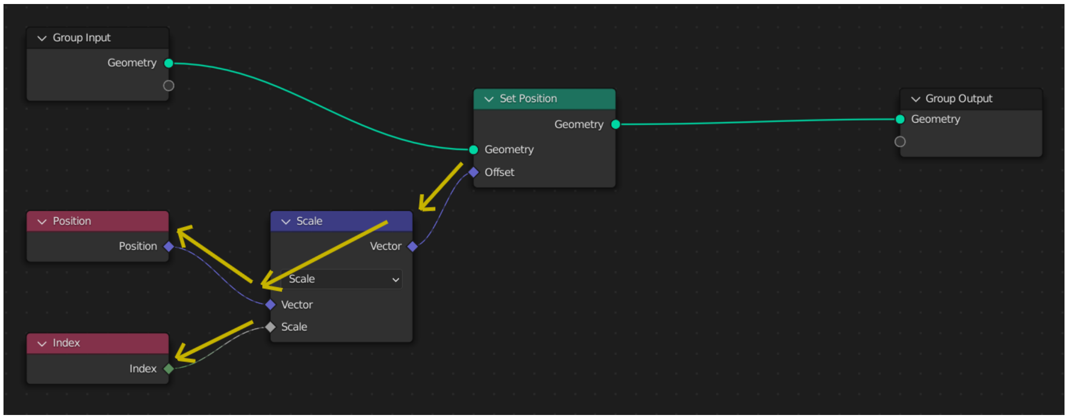
Input nodes provide data to the field evaluation process. By themselves, they mean nothing; they must be evaluated within the context of a data flow node (geometry) to actually output a value. Examples of input nodes are the built-in attribute input nodes like Position and ID, but also selection nodes like Endpoint Selection.

Field inputs may also come from other nodes that process geometry like the Distribute Points on Faces, in the form of Anonymous Attributes.

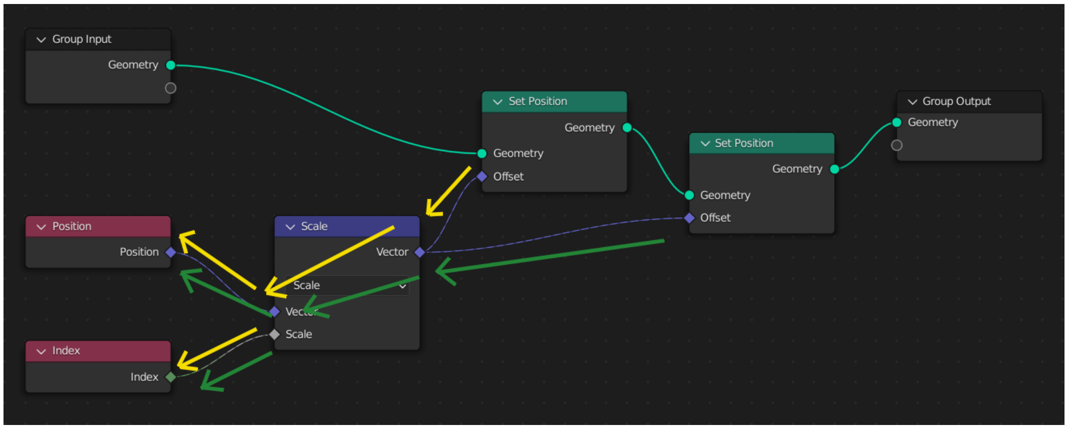
Field Context

All field nodes work in the context of the data flow node they are connected to. The context usually consists of a geometry component type and an attribute domain, so it determines what data is retrieved from input nodes.

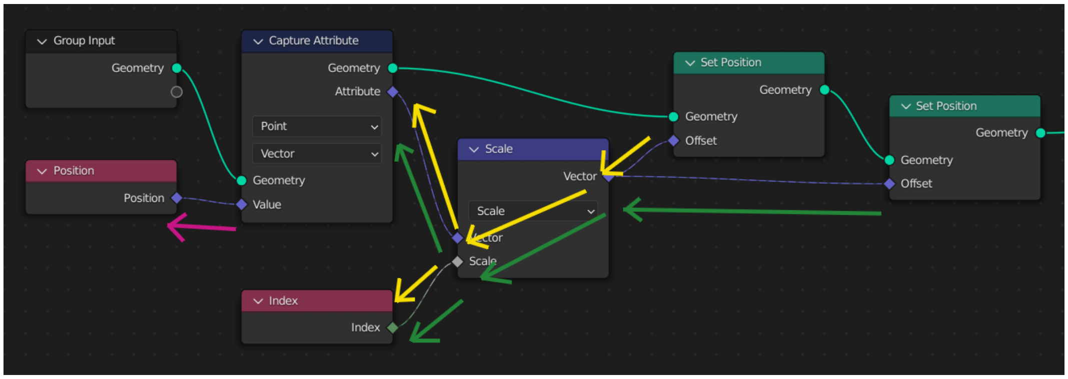
One common misunderstanding is that the same field node tree used in multiple places will output the same data. This is not necessarily true, because the field node tree will be evaluated for every data flow node, potentially retrieving data from a different or changed geometry.


Group Input
Geometry
Set Position
Geometry
Geometry
Offset
Group Output
Geometry
Position
Position
Scale
Vector
Scale
Vector
Scale
Index
Index

Here, the Set Position node's input field is evaluated once. To evaluate the field, the node traverses backwards to retrieve the inputs from the field input nodes.


Group Input
Geometry
Set Position
Geometry
Geometry
Offset
Set Position
Geometry
Geometry
Offset
Group Output
Geometry
Position
Position
Scale
Vector
Scale
Vector
Scale
Index
Index

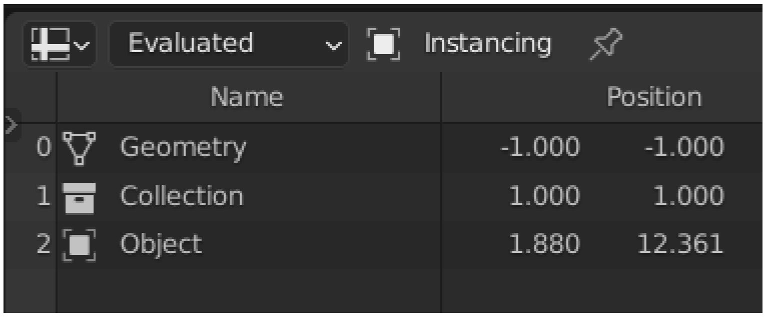
When a second Set Position node is added, the same field node tree is evaluated twice, once for each data flow node. At the second Set Position node, the results will be different since its geometry input will already have the changed position from the first node.


Group Input
Geometry
Capture Attribute
Geometry
Attribute
Point
Vector
Geometry
Value
Position
Position
Scale
Vector
Scale
Vector
Scale
Set Position
Geometry
Geometry
Offset
Set Position
Geometry
Geometry
Offset
Index
Index

However, often it's necessary to use the same field values even after changing the geometry. The Capture Attribute Node evaluates a field, copying the result to an anonymous attribute on the geometry.

Here, a Capture Attribute node stores a copy of the initial position. Notice that evaluating the field input of the Capture Attribute node is an entirely different step. Later on, the input fields to the Set Position nodes don't use the actual position, but the anonymous attribute copy of it.

Instances



	Name	Position	
0	Geometry	-1.000	-1.000
1	Collection	1.000	1.000
2	Object	1.880	12.361

The three types of instances.

In addition to storing real data like a mesh or a curve, objects can store instances, which themselves can reference more geometry, an object, or a collection. The purpose of instancing is to allow duplicating geometry and storing it in an object, without duplicating the actual data. This optimization allows render engines like Cycles to handle the same geometry data in many different locations better than when the data is duplicated.

Each instance keeps track of which geometry it corresponds to and how the instanced is transformed compared to it's source geometry. Instances can also store the `id` attribute, which is used to correct motion blur when instances move in an animation.

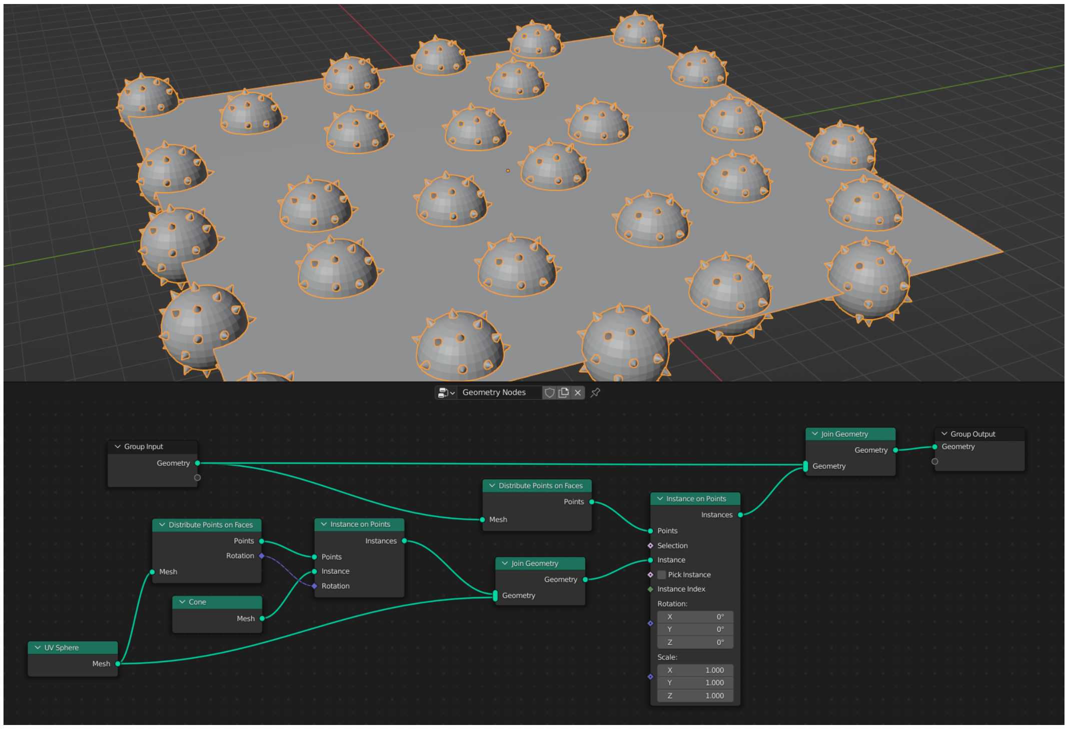
Instances can be created with geometry nodes using the Instance on Points Node.

Warning

Currently instancing from geometry nodes cannot be mixed with instancing from the Instancing panel in the property editor.

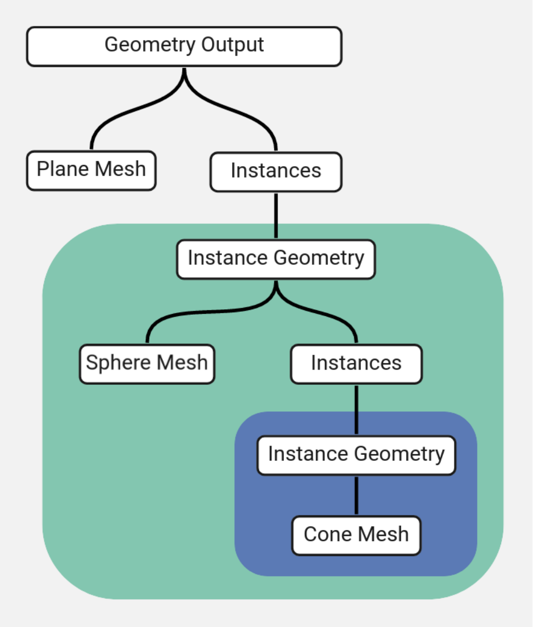
Nested Instancing

Since instances can store a geometry, and a geometry can contain instances, nested instancing is possible. In other words, it is possible to instance an instance, or even a collection of instances. For example, by default, the Instance on Points Node will create nested instances by instancing instances on the points of real geometry.


Geometry Nodes
Group Input
Geometry
Distribute Points on Faces
Points
Rotation
Mesh
Instance on Points
Instances
Points
Instance
Rotation
Cone
Mesh
UV Sphere
Mesh
Distribute Points on Faces
Points
Mesh
Join Geometry
Geometry
Geometry
Instance on Points
Instances
Points
Selection
Instance
Pick Instance
Instance Index
Rotation:
X 0°
Y 0°
Z 0°
Scale:
X 1.000
Y 1.000
Z 1.000
Join Geometry
Geometry
Geometry
Group Output
Geometry

A node group that creates nested instancing by chaining Instance on Points nodes.

Here, nested instancing is used to distribute geometry that contains both a mesh and instances. The output geometry contains a “real” mesh and a group of instances. Each instance contains a sphere mesh and many instances of a cone geometry.


Geometry Output
Plane Mesh
Instances
Instance Geometry
Sphere Mesh
Instances
Instance Geometry
Cone Mesh

The tree of instanced geometry for the example above.

What makes this method helpful is the output geometry only contains three unique meshes: the plane, the sphere, and the cone. This would make the performance much better if the meshes were more complicated.

Warning

Only eight levels of nested instancing are supported for rendering and viewing in the viewport. Though deeper trees of instances can be made inside geometry nodes, they must be realized at the end of the node tree.

Realizing Instances

The term “realizing” instances referes to converting the instances into unique geometry. When instances are realized they will take up more memory and manipulation to geometry will have to be processed individually rather the once per instancing geometry.

To realize instance use the Realize Instances Node.

Instance Processing

Almost all nodes that process geometry do so by processing each unique geometry separately rather than realized geometry. For example, if a Subdivision Surface Node was placed at the end of the example above, it would only have to subdivide three meshes, rather than each instance of a mesh. Another important example is processing with the output of the String to Curves Node, where each unique character only has to be processed once.

This method can improve performance a lot, but it means that the result of an operation will be the same for every instance of a certain geometry. In order to have unique results for every instance, the Realize Instances Node node can be used.

Baking

Baking allows saving and loading intermediate geometries. Baking parts of the node tree can be used for better performance.

The data format used to store geometry data is not considered to be an import/export format. Volume objects, however, are saved using the OpenVDB file format which can be used interoperably.

Data can be baked using two methods:

- Bake Node – used to bake any portion of the node tree.
- Simulation Zone Baking – used to bake animations where the result of one geometry state can influence the next state.

Data is not stored to disk until the Bake operation is ran.

Important

- Blend-files must be saved to a disk before data can be baked.
- It’s not guaranteed that data written with one Blender version can be read by another Blender version.

Bake Geometry Node

Bakes the simulations and bake nodes in all modifiers for the selected objects.

Tip

When data is baked the number of baked frames is displayed above the simulation zone or bake nodes.

Data-Block References

Reference

Editor:: Geometry Node Editor
Panel:: *Sidebar ‣ Node ‣ Data-Block References*

Baked geometries that reference other data-blocks such as materials are listed here.

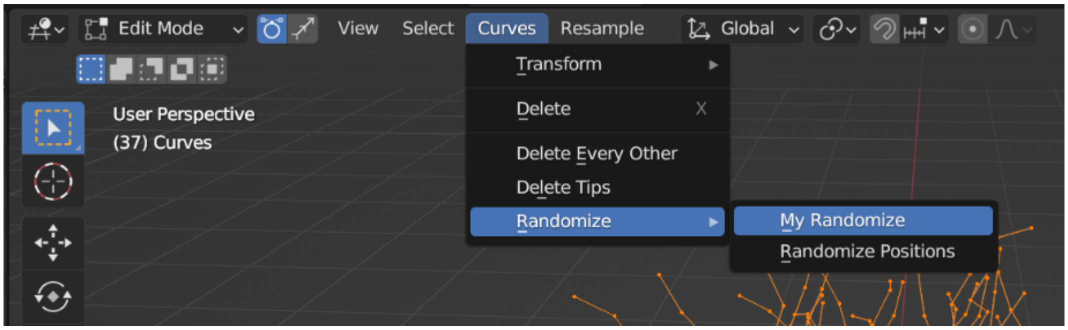
This panel allows changing these references after the data has been baked.

Note

Currently only material data-blocks are supported.

Node-Based Tools

Geometry Nodes can be used to expand the core functionality of Blender via node-group-defined tools. They can be shared as any regular node group assets.


Edit Mode
View
Select
Curves
Resample
Global
User Perspective
(37) Curves
Transform
Delete
X
Delete Every Other
Delete Tips
Randomize
My Randomize
Randomize Positions

Node group tools integrated in the Selection menu.

Tool Context

The way to create Node-based tools is by switching the Geometry Nodes editor context to *Tool*.

New node groups created in the tool context will be enabled as Tool by default, although users still need to set them as Assets if they want to share them (see below).

Note

The Inspection features are not supported in the *Tool* context: Viewer Node and Socket Inspection.

Asset

For a node group to be shared as a tool, it has to be an Asset first. The asset catalog is used to determine on which menus the node group will show, similar to the regular node group assets. If the catalog name matches an existing menu, the tool will be added to the end of it. Assets that have no catalog assigned to them, or local tools are exposed in the “Unassigned” menu.

The asset options need to be set on the Asset Browser.

Tool Settings

The node group inputs will be exposed as in the Adjust Last Operation panel.

Supported Modes & Data-Types

Node groups must specify which modes and object types they support. This helps to determine where the tool is available in the user interface. These properties can be configured in popover menus in the Geometry Node editor when in the *Tool* context.

Currently only Object, Edit, and Sculpting modes are supported, and only for the Mesh and Hair Curves object types.

Important

For mesh objects, Shape Keys are not supported. Operating a node tool on a mesh with shape keys will remove the shape key data.

Tool-specific Nodes

The following nodes are only supported in the *Tool* context:

- 3D Cursor Node
- Mouse Position Node

- Viewport Transform Node
- Active Element Node
- Selection Node
- Set Selection Node
- Face Set Node
- Set Face Set Node

Note

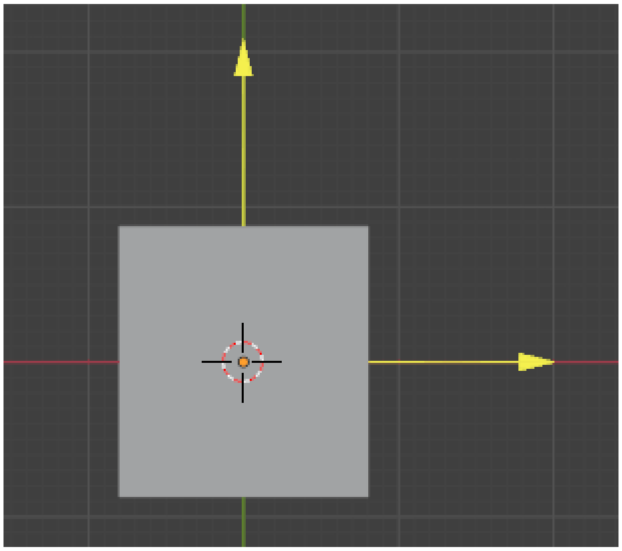
The Self Object node returns the Active object when inside a *Tool* node group.

Non-supported Nodes

These nodes are only supported in the *Modifier* context:

- Simulation Zone
- Viewer Node

Gizmos

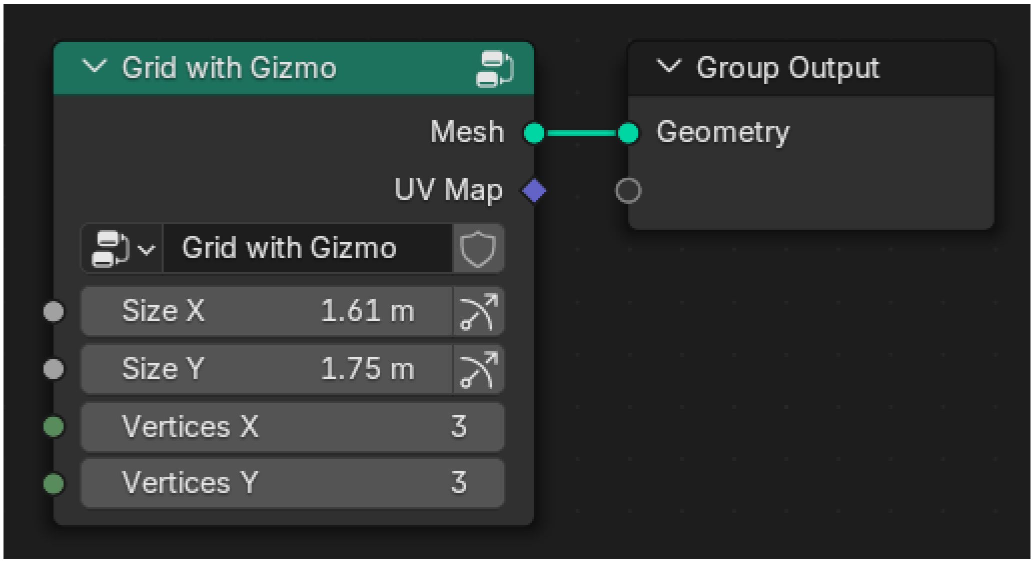


Grid with gizmo in the 3D viewport.

Gizmos allow changing inputs of Geometry Nodes directly in the 3D viewport. This is often more intuitive than controlling the inputs in the modifier or node editor.

Using Gizmos

Node inputs that can be controlled with a gizmo have an additional icon. The gizmos of that node are shown if it is selected. Clicking on the icon pins the gizmo, so that it is shown even if the node is not selected. Modifying the gizmo in the 3D viewport, modifies the value in the socket.

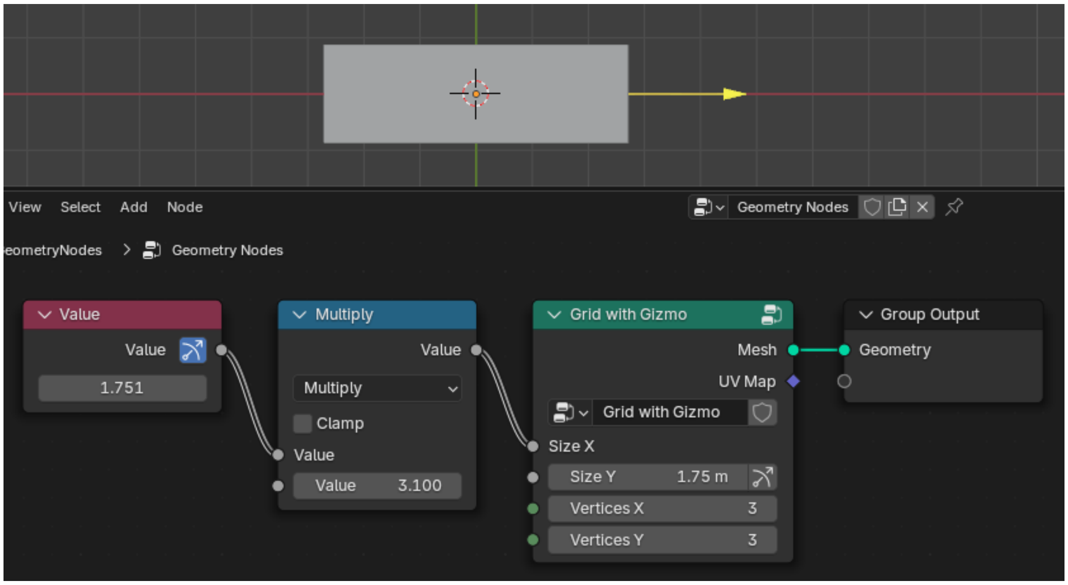

Grid with Gizmo
Mesh
UV Map
Group Output
Geometry
Grid with Gizmo
Size X 1.61 m
Size Y 1.75 m
Vertices X 3
Vertices Y 3

Example of a node group that has gizmos.

Note

Built-in nodes do not have their own gizmos yet. It’s possible to create node groups that have gizmos though.

Gizmos are often automatically propagated when an input socket with a gizmo is linked. The gizmo then controls the value that it is propagated to, instead of the input of the node group directly. Not all nodes support propagating gizmos, but many math operations do. A double link indicates that the propagation was successful.


View
Select
Add
Node
Geometry Nodes
eometryNodes
Geometry Nodes
Value
Multiply
Grid with Gizmo
Group Output
Mesh
Geometry
UV Map
Size X

Value 1.751

Multiply

Clamp

Value

Value 3.100

Grid with Gizmo

Size Y 1.75 m

Vertices X 3

Vertices Y 3

The gizmo is propagated from the Size X input socket to the Value node.

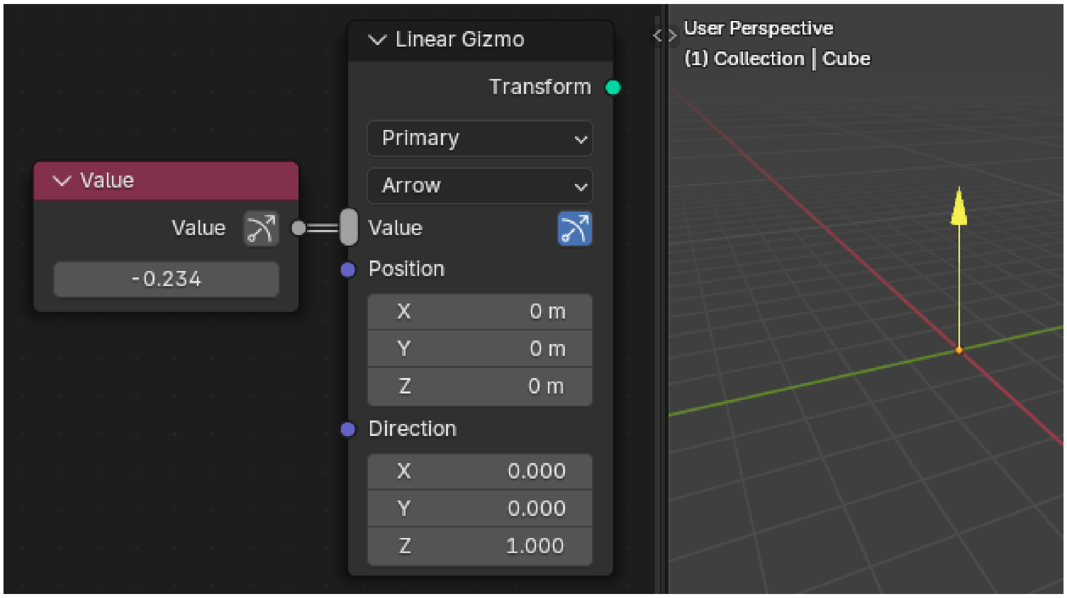
Gizmos can also be propagated to Group Inputs, in which case they are also available on the parent group node. If the current group is used by a modifier directly, the gizmo will also be available on the modifier. Gizmos that are propagated to the modifier always show when the modifier is active, independent of whether any node is visible or selected.

Creating Custom Gizmos

Adding custom gizmos to a node group that generates or modifies geometry can make it more convenient to use.

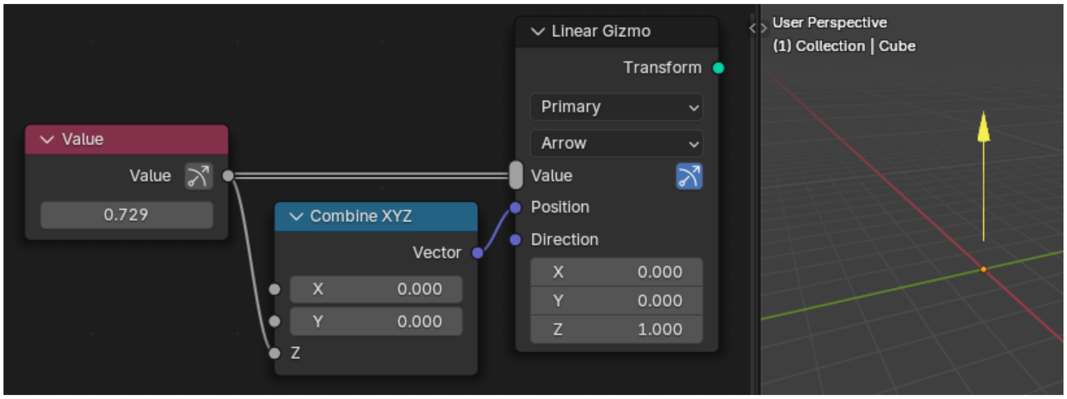
To add a gizmo to a node group, one has to use one of the gizmo nodes. The main aspect that makes gizmos unintuitive at first is that there is a **bidirection dependency**: changing the gizmo position changes the controlled value and vice versa.

The most simple custom gizmo setup is shown below. The Linear Gizmo node adds a gizmo that is drawn in the 3D viewport. The gizmo controls the value that is plugged into it. When trying this, you will notice that the gizmo always jumps back to the origin while the value is still changed. That is because the Position of the gizmo node does not dependend on the value yet.


Linear Gizmo
Transform
Primary
Arrow
Value
Position
X 0 m
Y 0 m
Z 0 m
Direction
X 0.000
Y 0.000
Z 1.000
Value
Value
-0.234
User Perspective
(1) Collection | Cube

The simplest custom gizmo setup.

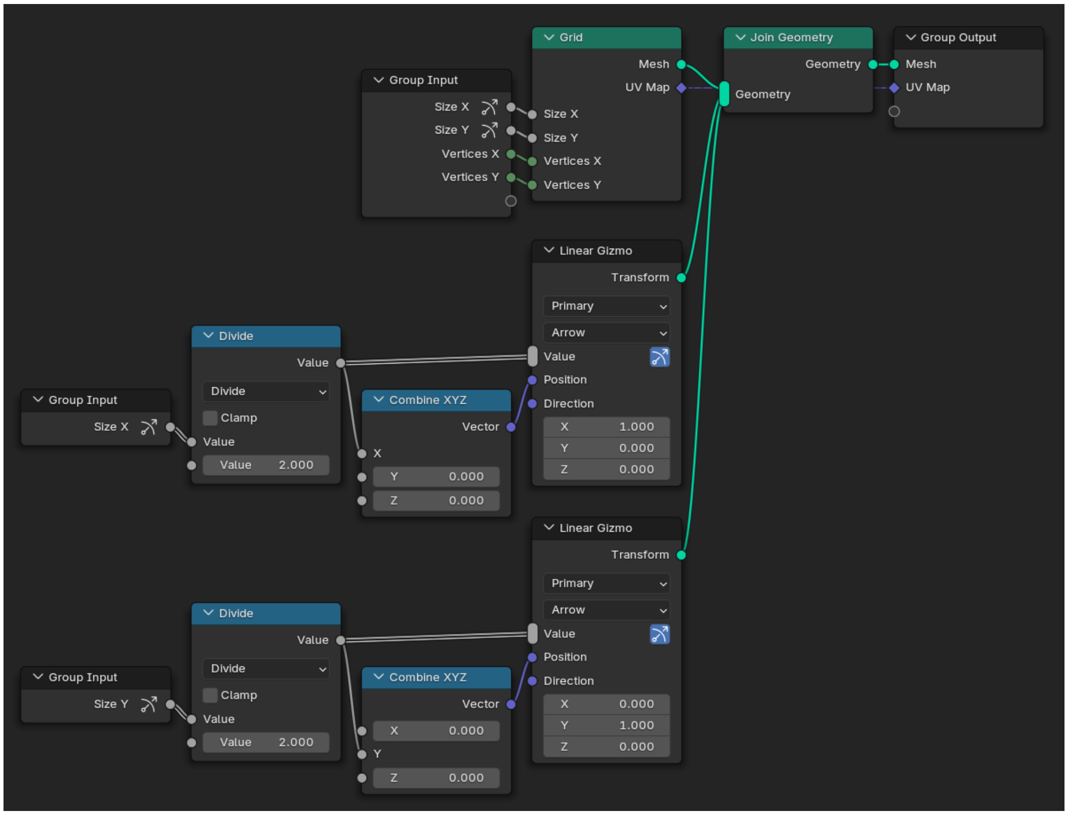
When the gizmo position is made dependent on the value, the gizmo works more than one would expect. It now also works in both directions: changing the value moves the gizmo and moving the gizmo changes the value.


Linear Gizmo
Transform
Primary
Arrow
Value
Position
Direction
X 0.000
Y 0.000
Z 1.000
Value
Value
0.729
Combine XYZ
Vector
X 0.000
Y 0.000
Z
User Perspective
(1) Collection | Cube

Simple gizmo setup where the gizmo position depends on the controlled value.

Multiple values can be plugged into the Value input of gizmo nodes at once. In that case, all these values are modified at the same time when moving the gizmo. Multiply or divide nodes can be used if the values should change at different rates.

The Transform output of gizmo nodes should be joined into the geometry that the gizmo controls. This helps Blender to understand that the gizmos should be transformed together with the geometry later on.


Grid
Mesh
UV Map
Size X
Size Y
Vertices X
Vertices Y
Join Geometry
Geometry
Geometry
Group Output
Mesh
UV Map
Group Input
Size X
Size Y
Vertices X
Vertices Y
Linear Gizmo
Transform
Primary
Arrow
Value
Position
Direction
X 1.000
Y 0.000
Z 0.000
Divide
Value
Divide
Clamp
Value
Value 2.000
Group Input
Size X
Combine XYZ
Vector
X
Y 0.000
Z 0.000
Linear Gizmo
Transform
Primary
Arrow
Value
Position
Direction
X 0.000
Y 1.000
Z 0.000
Divide
Value
Divide
Clamp
Value
Value 2.000
Group Input
Size Y
Combine XYZ
Vector
X 0.000
Y
Z 0.000

Example showing how to add simple gizmos to the built-in Grid node.

Note

Generally, it is possible to build the entire node group functionality first and to add gizmos afterwards.

Attribute Nodes

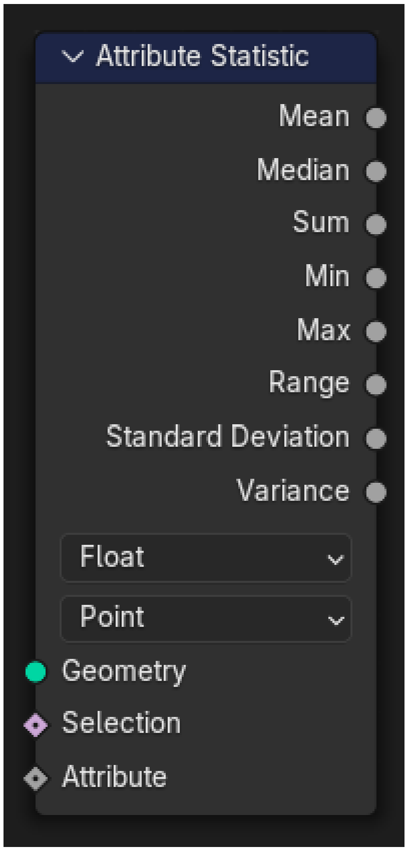
Nodes for working with data stored per object element, e.g. vertex groups.

- Attribute Statistic Node
- Domain Size Node

- Blur Attribute Node
- Capture Attribute Node
- Remove Named Attribute Node
- Store Named Attribute Node

Attribute Statistic Node

The *Attribute Statistic* node evaluates a field on a geometry and outputs a statistic about the entire data set.


Attribute Statistic
Mean
Median
Sum
Min
Max
Range
Standard Deviation
Variance
Float
Point
Geometry
Selection
Attribute

Inputs

Geometry
: Standard geometry input.

Selection
: A boolean field input for each element indicating whether to include its value in the statistics result If the boolean is false, the corresponding value from the *Attribute* input will be ignored.

Attribute
: The attribute field to query a statistic from.

Properties

Data Type

Float::	The output will be a single floating-point value.
Vector::	The output will be a vector of three floating-point values. All calculations are elementwise.

Domain

The attribute domain used for the statistics and to evaluate the input *Attribute* field on.

Outputs

Mean

The average value of all data.

Median

The median value of all data.

Sum

The sum value of all data.

Min

The min value of all data.

Max

The max value of all data.

Range

The difference between the max and min value.

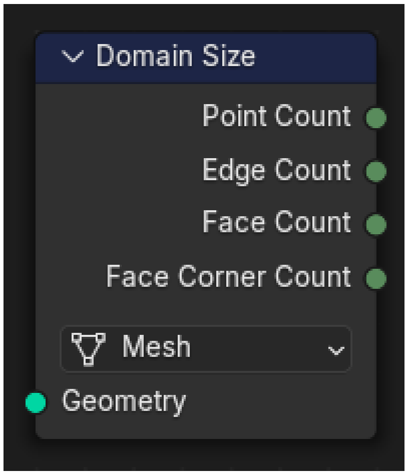
Standard Deviation

How much values differ from the mean. A low standard deviation indicates that the values are grouped tightly together at the mean. A high standard deviation indicates that the values are spread out over a large range.

Variance

The variance of all data, defined as the square of the standard deviation.

Domain Size Node


Domain Size
Point Count
Edge Count
Face Count
Face Corner Count
Mesh
Geometry

The *Domain Size* outputs the size of an attribute domain on the selected geometry type, for example, the number of edges in a mesh, or the number of points in a point cloud.

For more information about attribute domains, see the geometry attributes page.

Inputs

Geometry
: Standard geometry input.

Properties

Component
: Which geometry type to retrieve the domain sizes from.

Outputs

Point Count
: The size of the *Point* domain on any supported geometry.

Edge Count
: The size of the *Edge* domain on meshes.

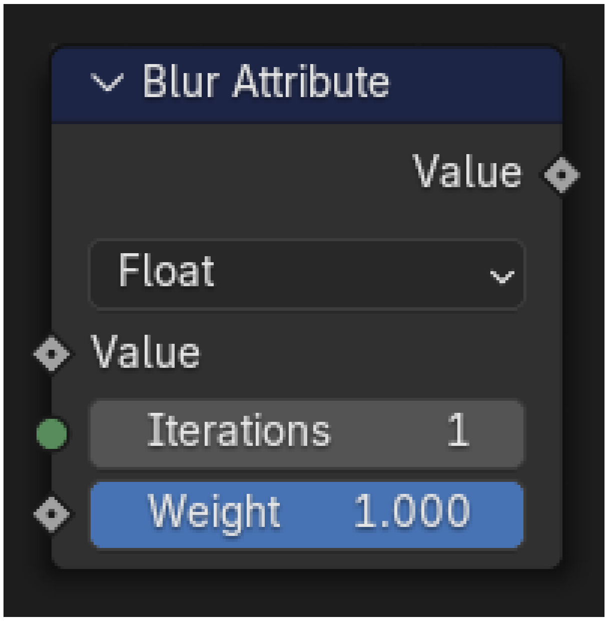
Face Count
: The size of the *Face* domain on meshes.

Face Corner Count
: The size of the *Face Corner* domain on meshes.

Spline Count
: The size of the *Spline* domain on curves.

Instance Count
: The number of top-level instances in the geometry. Nested instances are not considered.

Blur Attribute Node



The *Blur Attribute* node smooths attribute values between neighboring geometry elements.

The goal of each step is mixing values of each element with its neighbors. The weight for element is factor for multiplying all neighbor's values before accumulating them as new primitive value.

Blurring will only work with certain geometry types and attribute domains. Therefore, the attribute can only be affected on the Meshes and Curves components.

The domains this node works on is based on the field context of the node's evaluation. Only domains with explicit relations with their neighbors will work with this node. Explicit relations for correct blurring are vertices, edges, and faces of meshes, and curve control points.

Note

Blurring of face corner attributes is not handled by this node, because the ideal behavior for mixing face corner values is not clear.

All attribute data types are supported except for boolean attributes.

Inputs

Value
: The immediate value of each primitive to blur.

Iterations
: Number of repetitions of mixing value with neighbors. Each iteration is independent. Until one iteration is completed, its results are not used as a source for next blurring.

Weight
: Weight of each primitive.

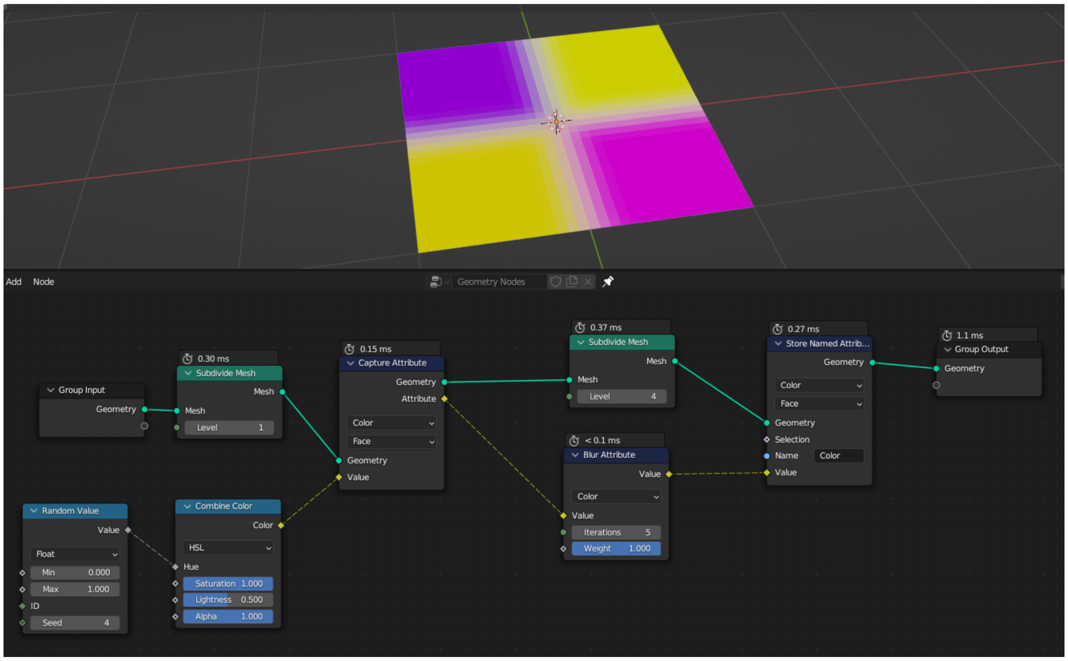
Properties

Data Type
: The data type used for the evaluated data.

Outputs

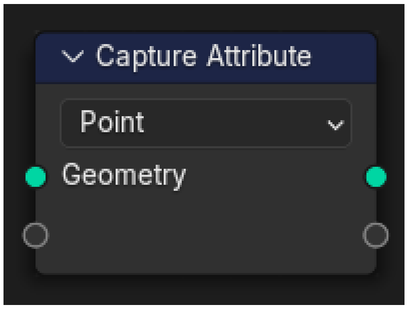
Value
: Values, mixed with neighbors defined number of times.

Examples


Add
Node
Geometry Nodes
Group Input
Geometry
0.30 ms
Subdivide Mesh
Mesh
Mesh
Level 1
0.15 ms
Capture Attribute
Geometry
Attribute
Color
Face
Geometry
Value
0.37 ms
Subdivide Mesh
Mesh
Mesh
Level 4
< 0.1 ms
Blur Attribute
Value
Color
Value
Iterations 5
Weight 1.000
0.27 ms
Store Named Attrib...
Geometry
Color
Face
Geometry
Selection
Name Color
Value
1.1 ms
Group Output
Geometry
Random Value
Value
Float
Min 0.000
Max 1.000
ID
Seed 4
Combine Color
Color
HSL
Hue
Saturation 1.000
Lightness 0.500
Alpha 1.000

Input is Mesh Plane. First Subdivide Mesh Node add some faces for capture color with Random Value Node used as hue in Combine Color Node on this. Now second Subdivide Mesh Node split each face on a lot of new. Each one new duplicates original attribute. Blur Attribute node mixes all attributes for each face. Due to this, the result is smoothed.

Capture Attribute Node



The *Capture Attribute* node stores the result of a field on a geometry, and outputs the data as a node socket so it can be used by other nodes.

The result is stored on the geometry just like a regular attribute with a name, but instead of referencing it with a name, it is retrieved whenever the socket is connected to the input of a node. Later on when evaluating the node tree, the attribute will be removed automatically if it is no longer used.

This node is essential because field input nodes like the Radius Node work in the context of the node they are connected to. Meaning that in order to pass data like `radius` to a geometry that doesn't have radius, an explicit node link with the output of this node must be used.

Note

Because this node stores an anonymous attribute in the geometry, it's essential to use the geometry output for further operations in the node tree. The anonymous attribute will not exist for any other geometry besides the output.

Inputs

Geometry
    Standard geometry input.

Capture Items
    The input field to evaluate. More fields can be added by dragging sockets into the blank socket or in the *Capture Items* panel. Items can be

renamed by `Ctrl-LMB` on the socket name or in the nodes *Properties* panel.

Properties

Domain
Which attribute domain to store the evaluated data on.

Capture Items

Reference

Menu:: *Sidebar ‣ Node ‣ Properties ‣ Capture Items*

Manages the input fields sockets of the node. Field sockets can be added, removed, reorganized, and renamed from the List View.

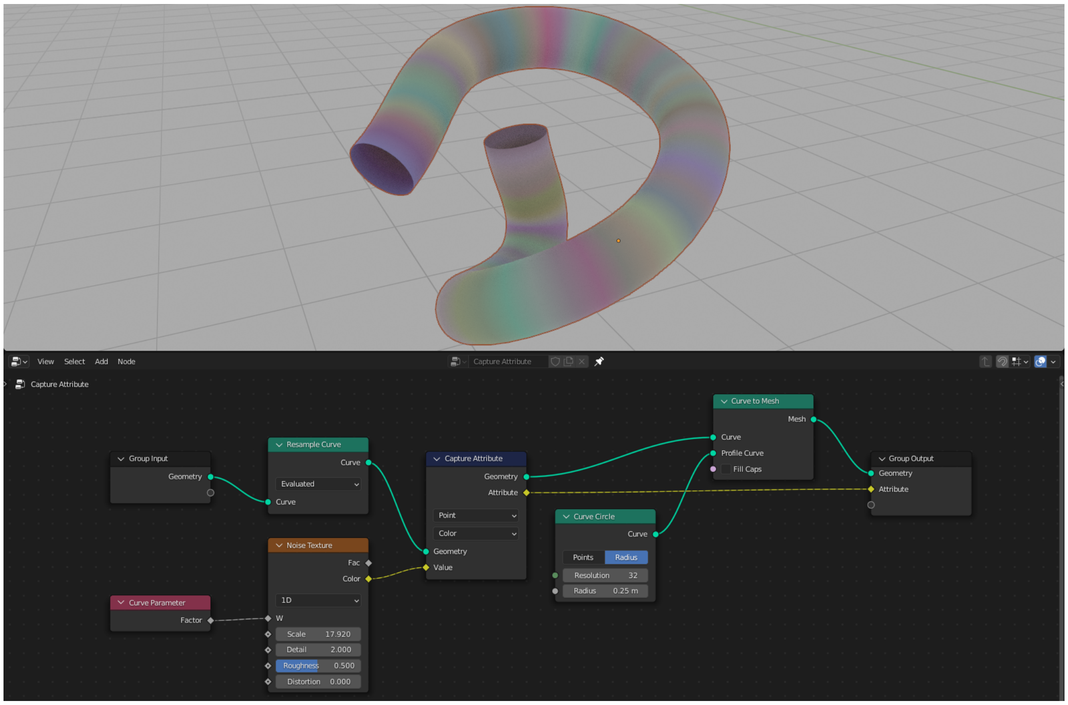
Data Type
The data type used for the evaluated data.

Outputs

Geometry
Standard geometry output.

Attribute
The result of the evaluated field, stored on the geometry.

Examples

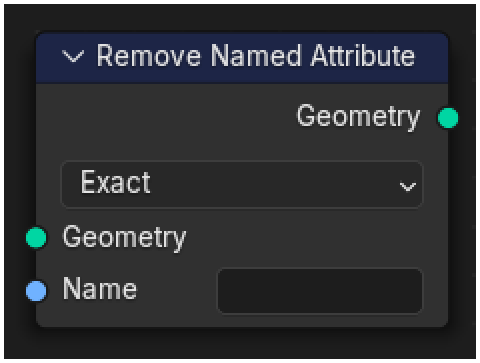

View
Select
Add
Node
Capture Attribute
Capture Attribute
Group Input
Geometry
Resample Curve
Curve
Curve
Capture Attribute
Geometry
Attribute
Geometry
Value
Curve to Mesh
Mesh
Curve
Profile Curve
Fill Caps
Group Output
Geometry
Attribute
Noise Texture
Fac
Color
W
Curve Parameter
Factor
Curve Circle
Curve
Points
Radius

Evaluated
Point
Color
1D
Scale 17.920
Detail 2.000
Roughness 0.500
Distortion 0.000
Resolution 32
Radius 0.25 m

Here, a noise texture is evaluated in along the path of the curve in one dimension and rendered with a shader. The capture node is required because the output of the Curve to Mesh Node does not have a “curve parameter”, since it is a mesh and not a curve. So, the Spline Parameter Node must be evaluated while the geometry is still a curve.

Internally, after the noise texture is evaluated on the curve, it is automatically copied to the mesh result of the Curve to Mesh node. This means that anywhere *Attribute* output of this node can be connected along the same stream of geometry nodes, the internal attribute will be available.

Remove Named Attribute Node



The *Remove Named Attribute* node deletes an attribute with a certain name from its geometry input. Any attribute that exists on geometry data will be automatically propagated when the geometry storing it is changed, which can be an expensive operation, so using this node can be a simple way to optimize the performance of a geometry node tree or even to lower the memory usage of the entire scene.

Almost all named attributes can be removed. For certain Built-In Attributes, removing it will mean that a default value will be used instead. For example, removing the cyclic attribute on curves means that all curves will be non-cyclic.

Inputs

Geometry
    Standard geometry input.

Name
    The name of the attribute to remove.

Properties

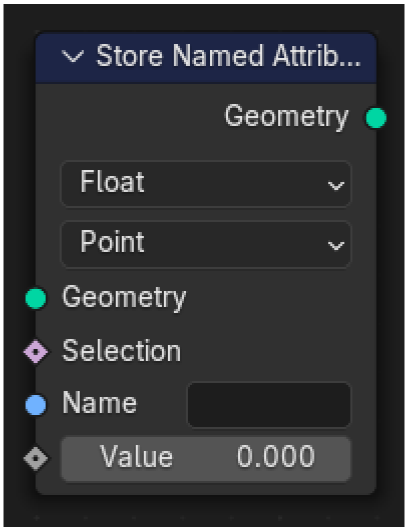
This node has no properties.

Outputs

Geometry

Standard geometry output.

Store Named Attribute Node



The *Store Named Attribute* node stores the result of a field on a geometry as an attribute with the specified name. If the attribute already exists, the data type and domain will be updated to the values chosen in the node. However, keep in mind that the domain and data type of Built-In Attributes cannot be changed.

Compared with the Capture Attribute Node, this node basically does the same thing, but the attribute gets a name instead of an anonymous reference. For reusing the data in the same node tree, the *Capture Attribute* node might be preferable since it does not create the chance for name conflicts in the input geometry.

Note

If the input geometry contains multiple geometry component types, the attribute will be created on each component that has the chosen domain.

Inputs

Geometry
: Standard geometry input.

Selection
: A boolean field input for each element indicating whether to store the attribute value for the given index. If the attribute does not exist, unselected parts are filled with zero values; otherwise, only the selected parts are filled.

Value
    The input field to evaluate.

Name
    The name to give the stored data.

Properties

Data Type
    The data type used for the evaluated data.

Domain
    Which attribute domain to store the evaluated data on.

Outputs

Geometry
    Standard geometry output.

Input Nodes

Nodes used mainly as input to other nodes.

- Constant
- Gizmo
- Scene

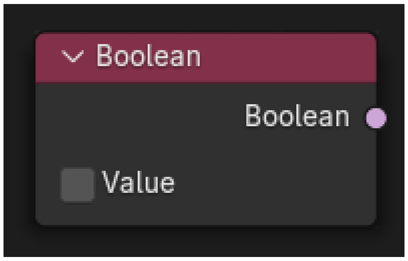
Input Constant Nodes

Nodes used mainly as input to other nodes.

- Boolean Node
- Color Node
- Image Node
- Integer Node
- Material Node
- Rotation Node
- String Node
- Value Node
- Vector Node

Boolean Node

The *Boolean* node provides a Boolean value.


Boolean
Boolean
Value

Inputs

This node has no input sockets.

Properties

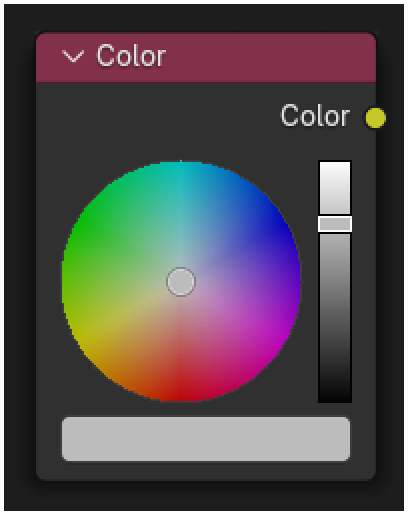
Single Boolean value (true/false).

Outputs

Boolean
    Standard Boolean output.

Color Node

The *Color* node outputs the color value chosen with the color picker widget.



Inputs

This node has no inputs.

Properties

This node has no properties.

Outputs

Color
    Color value indicated by the color picker widget.

Image Node

The *Image* node provides access to a image file which allows you to conveniently enter and switch images for multiple nodes in the tree.

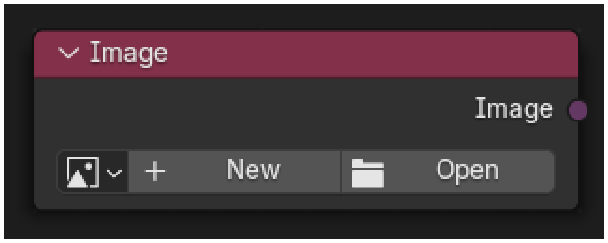


Image node.

See also

Image Info Node

Inputs

This node has no inputs.

Properties

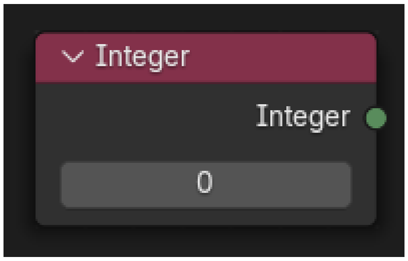
Image Data-Block
    The data-block selector to select an existing image or open an image from the file browser.

Outputs

Image
    The image file chosen from the data-block selector.

Integer Node

The *Integer* node provides an integer value.



Inputs

This node has no input sockets.

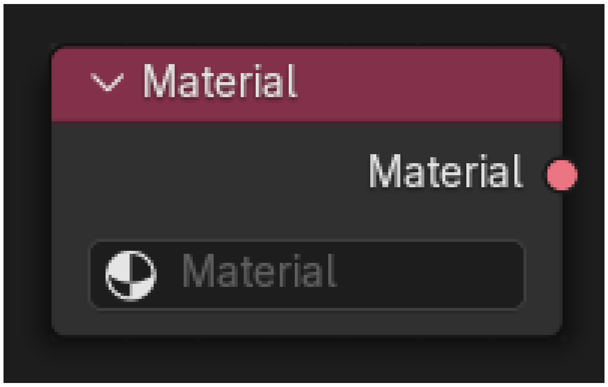
Properties

Single integer value.

Outputs

Integer
    Standard integer output

Material Node



The *Material* input node outputs a single material. It can be connected to other material sockets to make using the same material name in multiple places more convenient.

Tip

The Material node can also be added by dragging and dropping a material data-block into the node editor. This will add the node and select the dropped material in the Data-Block Menu.

Inputs

This node has no inputs.

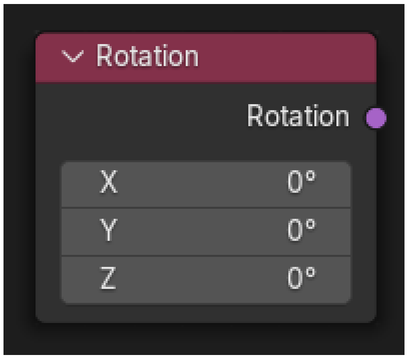
Properties

- Material

Output

Material
A reference to the selected material.

Rotation Node


Rotation
Rotation
X 0°
Y 0°
Z 0°

The *Rotation* input creates a rotation from euler rotation values.

Inputs

This node has no inputs.

Properties

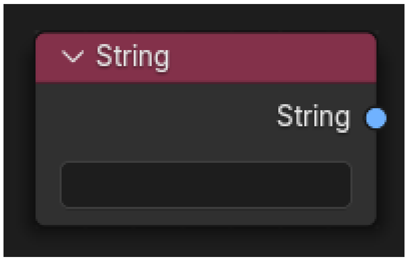
X, Y, Z
: The amount of rotation about each axes.

Output

Vector
: Standard rotation output.

String Node

The *String* input node creates a single string. It can be connected to attribute name sockets to make using the same attribute name in multiple places more convenient.



Inputs

This node has no inputs.

Properties

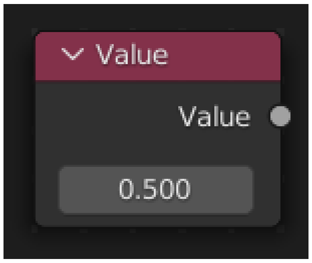
- String

Output

String
    Standard string output.

Value Node

The *Value Node* is a simple node to input numerical values to other nodes in the tree.



Inputs

This node has no input sockets.

Properties

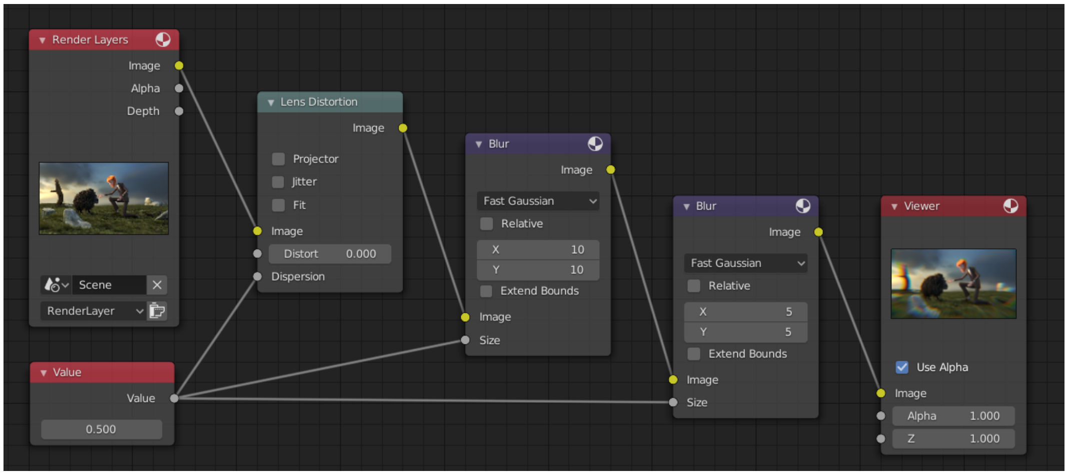
Single numerical value (floating-point).

Outputs

Value
  The value set in the node properties.

Example

In the following example the *Value Node* is used to control multiple values at once, this makes the node a useful organizational tool.


Render Layers
Image
Alpha
Depth
Scene
RenderLayer
Lens Distortion
Image
Projector
Jitter
Fit
Image
Distort 0.000
Dispersion
Blur
Image
Fast Gaussian
Relative
X 10
Y 10
Extend Bounds
Image
Size
Blur
Image
Fast Gaussian
Relative
X 5
Y 5
Extend Bounds
Image
Size
Viewer
Use Alpha
Image
Alpha 1.000
Z 1.000
Value
Value
0.500

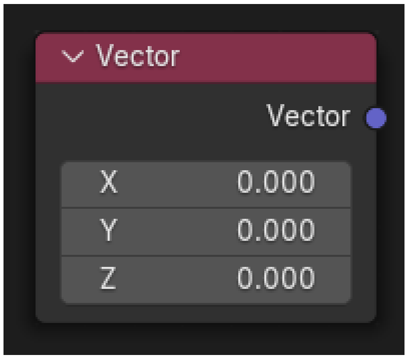
Example of the *Value* node.

Tip

From this you can also make different values proportional to each other by adding a Math Node in between the different links.

Vector Node

The *Vector* input node creates a single vector.



X 0.000
Y 0.000
Z 0.000

Inputs

This node has no inputs.

Properties

- X
- Y
- Z

Output

Vector
Standard vector output.

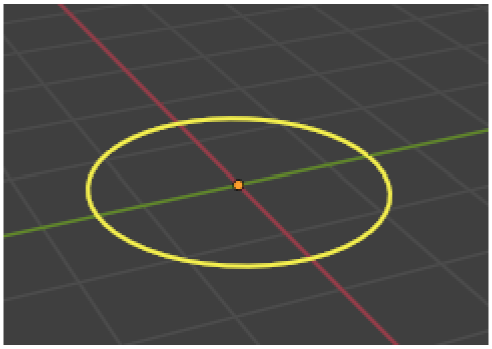
Gizmo Nodes

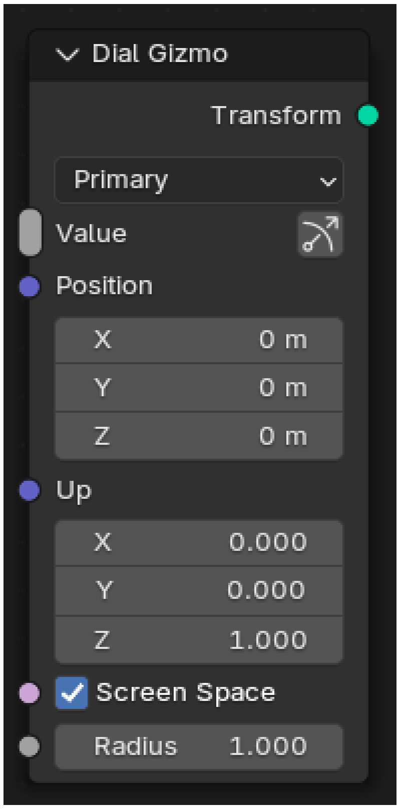
See this for a general guide on how to use gizmo nodes.

- Dial Gizmo
- Linear Gizmo
- Transform Gizmo

Dial Gizmo

The *Dial Gizmo* node is ideal for creating gizmos that control angles.





Dial Gizmo
Transform
Primary
Value
Position
X 0 m
Y 0 m
Z 0 m
Up
X 0.000
Y 0.000
Z 1.000
☑ Screen Space
Radius 1.000

Inputs

Value
: Special gizmo value socket. Everything that linked into this socket will be modified when the gizmo is rotated.

Position
: Position of the gizmo in the local space of the object.

Up
: Up or normal direction of the gizmo in the viewport.

Screen Space

If enabled, the gizmo will always have the same size in the viewport independent of the zoom level. This affects the meaning of the radius input.

Radius

In screen space mode, this is a factor on top of the default radius. Otherwise, this is the radius of the gizmo in Blender units.

Properties

Color

Controls which theme color is used for this gizmo.

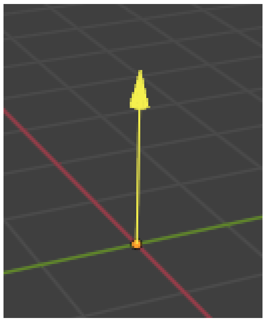
Outputs

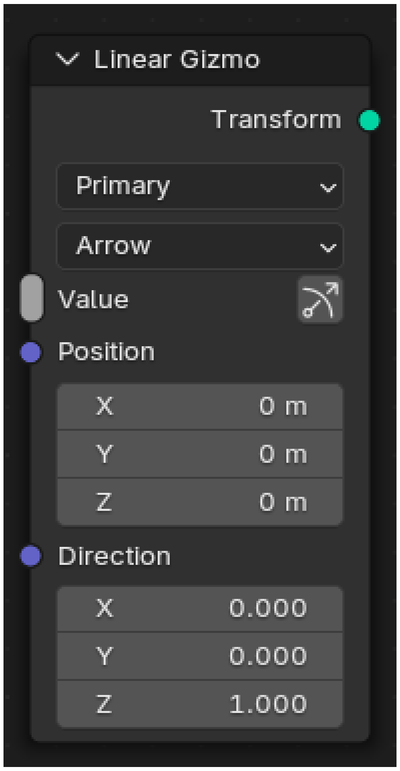
Transform

Should be joined into the geometry that is controlled by this gizmo.

Linear Gizmo

The *Linear Gizmo* node provides the most widely applicable gizmo. It can e.g. be used to control the height of something.





Linear Gizmo

Transform

Primary

Arrow

Value

Position

X 0 m

Y 0 m

Z 0 m

Direction

X 0.000

Y 0.000

Z 1.000

Inputs

Value
: Special gizmo value socket. Everything that linked into this socket will be modified when the gizmo is moved.

Position
: Position of the gizmo in the local space of the object.

Direction
: Specifies the direction in with the gizmo points or is moved.

Properties

Color
: Controls which theme color is used for this gizmo.

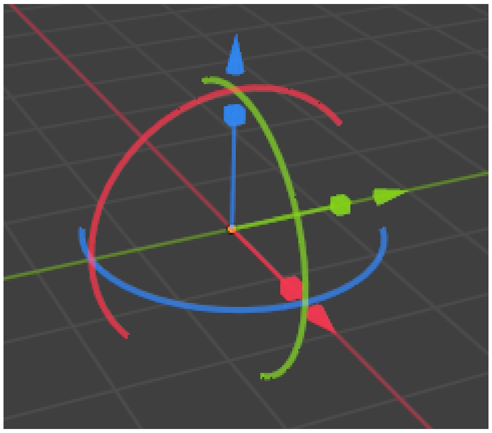
Draw Style
: Allows choosing between different styles of the gizmo.

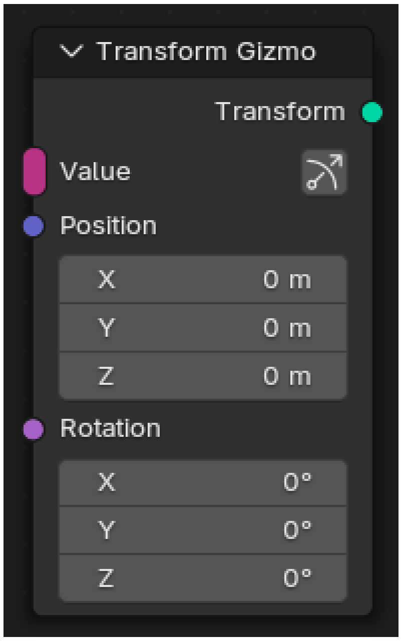
Outputs

Transform
: Should be joined into the geometry that is controlled by this gizmo.

Transform Gizmo

The *Transform Gizmo* node provides a compound gizmo that can control a position, rotation and scale.





Transform Gizmo

Transform

Value

Position

X 0 m

Y 0 m

Z 0 m

Rotation

X 0°

Y 0°

Z 0°

Inputs

Value
: Special gizmo value socket. Everything that linked into this socket will be modified when the gizmo is modified.

Position
: Position of the gizmo in the local space of the object.

Rotation
: Local orientation of the gizmo.

Note

The rotation input is ignored by the 3D viewport if the transform orientation is set to global.

Properties

The node has properties in the sidebar which allow disabling parts of the gizmo. This can be useful when e.g. controlling only a translation or only a rotation.

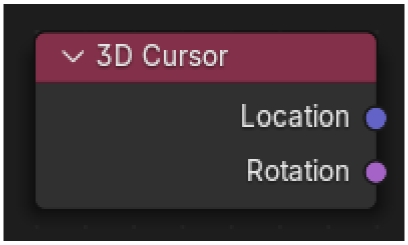
Outputs

Transform
    Should be joined into the geometry that is controlled by this gizmo.

Input Scene Data Nodes

- 3D Cursor Node
- Active Camera Node
- Collection Info Node
- Image Info Node
- Is Viewport Node
- Mouse Position Node
- Object Info Node
- Scene Time Node
- Self Object Node
- Viewport Transform Node

3D Cursor Node



The *3D Cursor* node outputs the position and orientation of the 3D cursor in the scene.

Note

This node can only be used in the Tool context.

Inputs

This node has no inputs.

Properties

This node has no properties.

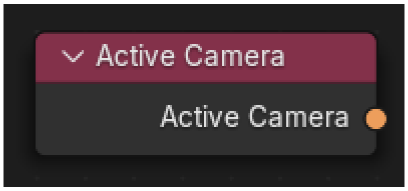
Outputs

Location
: The position of the 3D cursor.

Rotation
: The orientation of the 3D cursor as a standard rotation value.

Active Camera Node

The *Active Camera* node outputs the scene's current active camera.


Active Camera
Active Camera

Inputs

This node has no inputs.

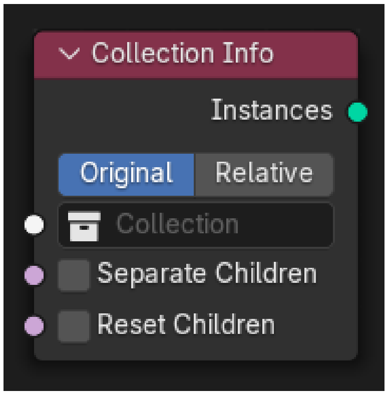
Properties

This node has no properties.

Outputs

Active Camera
    Current active camera.

Collection Info Node



The *Collection Info* node gets information from collections. This can be useful to control parameters in the geometry node tree with an external collection.

Tip

A *Collection Info* node can be added quickly by dragging a collection into the node editor.

Inputs

Collection
: Collection to get the properties from.

Separate Children
: Output each child of the collection as a separate instance. The list of instances will be sorted alphabetically with the objects and child collections sorted together. This can be used with the *Pick Instance* option in the Instance on Points Node to choose between collection children at each point.

Reset Children
: Remove the transform of each of the collection's children when converting them to instances. This is useful in order to keep child objects visually separate in the viewport, while keeping every instance located directly at the location of the point it was added for.

Properties

Transform Space

The transformation of the output instances. The instances are transformed, but not the geometry of the collection in them.

Original:: Output the instances relative to the collection offset.

Relative:: Join the input collection instances with the modified object as geometry, maintaining the relative position between the objects in the scene.

Outputs

Instances

Instances of the collection in world space with all modifiers applied and represented as geometry in instances.

Image Info Node

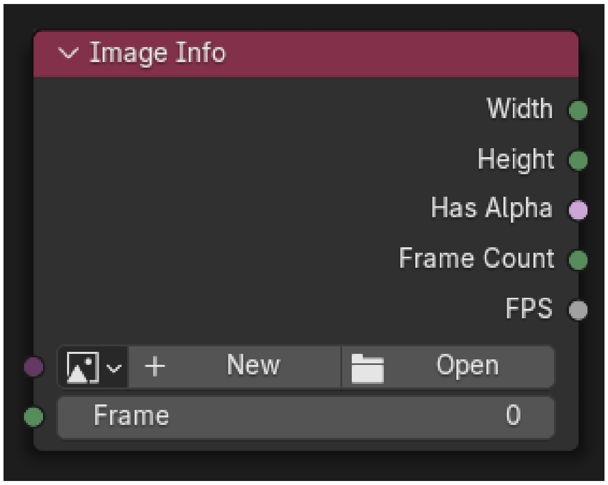


Image Info
Width
Height
Has Alpha
Frame Count
FPS
New
Open
Frame
0

The *Image Info* node gets information from image and animation. This can be useful to generate parameters in the geometry node for arbitrary images. Image information can be either general or frame-specific.

Inputs

Image
: Source image to get parameters from.

Frame
: Frame index for frame-specific outputs.

Properties

This node has no properties.

Outputs

Width
: The number of pixels along the X axis. Specific to each frame.

Height
: The number of pixels along the Y axis. Specific to each frame.

Has Alpha

Whether the transparency channel be different from 1 for the pixels of this image frame. Specific to each frame.

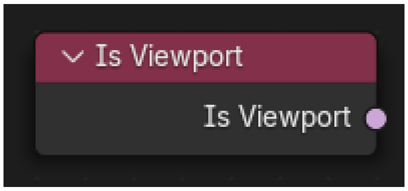
Frame Count

The number of frames in an image or video frame sequence. For a static image, always 1.

FPS

The number of frames per second. For static image is always 0.

Is Viewport Node



The *Is Viewport* node outputs true when geometry nodes are evaluated for the viewport. For the final render the node outputs false.

Inputs

This node has no inputs.

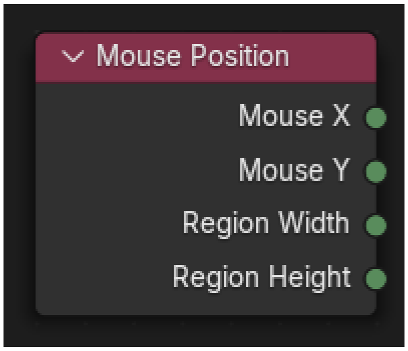
Properties

This node has no properties.

Outputs

Is Viewport
  Boolean value that indicates whether geometry nodes are evaluated for preview.

Mouse Position Node


Mouse Position
Mouse X
Mouse Y
Region Width
Region Height

The *Mouse Position* node returns information about the mouse cursor such as its position and the region's dimensions.

Tip

When using this node, enable Wait for Click. to wait for a mouse click input (`LMB`) before running the operator from a menu.

Note

This node can only be used in the Tool context.

Inputs

This node has no inputs.

Properties

This node has no properties.

Outputs

Mouse X

The region-space mouse X location, in pixels, increasing from 0 at the left.

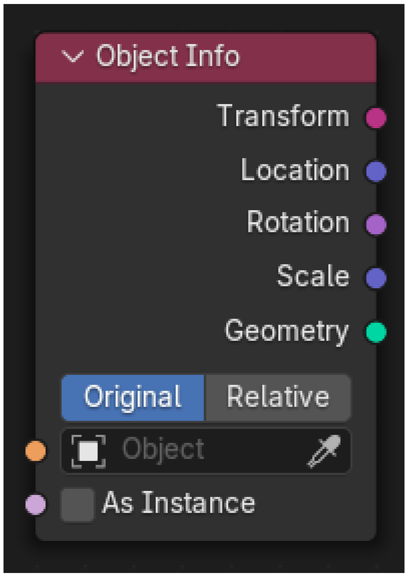
Mouse Y

The region-space mouse Y location, in pixels, increasing from 0 at the bottom.

Region Width
: The total X size of the region in pixels.

Region Height
: The total Y size of the region in pixels.

Object Info Node


Object Info
Transform
Location
Rotation
Scale
Geometry
Original
Relative
Object
As Instance

The *Object Info* node gets information from objects. This can be useful to control parameters in the geometry node tree with an external object, either directly by using its geometry, or via its transformation properties.

An *Object Info* node can be added quickly by dragging an object into the node editor.

Inputs

Object
: Object to get the properties from.

As Instance
: Output the entire object as single instance instead of realized geometry. This allows instancing non-geometry object types, because the output will contain an instance of the object.

Properties

Transform Space
: The transformation of the vector and geometry outputs.

Original::	Output the geometry relative to the input object transform, and the location, rotation and scale relative to the world origin.

Relative::	Bring the input object geometry, location, rotation and scale into the modified object, maintaining the relative position between the two objects in the scene.

Outputs

Transform
: Transformation Matrix containing the location, rotation and scale of the object.

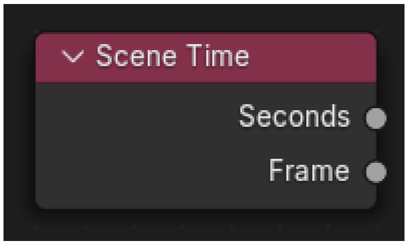
Location
: Location of the object in world space.

Rotation
: Rotation of the object in world space.

Scale
: Scale of the object in world space.

Geometry
: Geometry of the object in world space with all its modifiers applied.

Scene Time Node



The *Scene Time* node outputs the current time in the scene's animation in units of seconds or frames.

Inputs

This node has no inputs.

Properties

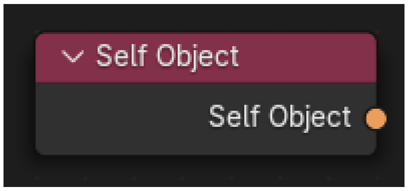
This node has no properties.

Outputs

Seconds
    Current scene time in seconds.

Frames
    Current scene frame. As an input in geometry nodes, this output may also output non-round numbers, in order to support higher quality motion blur.

Self Object Node



The *Self Object* node outputs the object that contains the geometry nodes modifier currently being executed. This can be used to retrieve the original transforms.

When evaluated in the Tool context, this node returns the Active object.

Note

The geometry cannot be retrieved from this object with the Object Info Node, since its final geometry is still being evaluated.

Inputs

This node has no inputs.

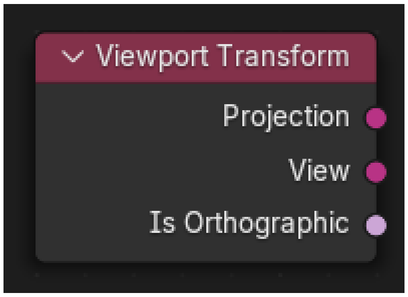
Properties

This node has no properties.

Outputs

Self Object
    The object currently being evaluated.

Viewport Transform Node


Viewport Transform
Projection
View
Is Orthographic

The *Viewport Transform* node retrieves the view direction and location of the 3D Viewport.

Note

This node can only be used in the Tool context.

Inputs

This node has no inputs.

Properties

This node has no properties.

Outputs

Projection
: The 3D Viewport's perspective or orthographic projection matrix.

View
: The view direction and location of the 3D viewport.

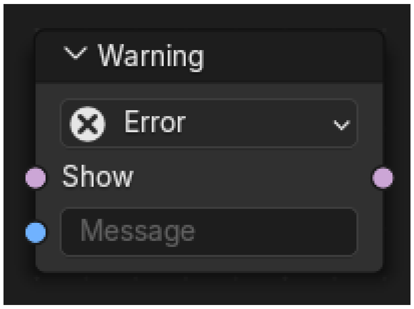
Is Orthographic
: Whether the viewport is using orthographic projection.

Output Nodes

Nodes that output to outside the node tree.

- Warning Node
- Viewer Node

Warning Node



Outputs a custom message that can be referenced in the Warnings panel of the Geometry Nodes Modifier.

This allows node groups to communicate expectations about input values.

By default, warnings are propagated through all parent node groups. However, this can be controlled using the Warning Propagation setting on each node.

Inputs

Show

Present the warning in the Warnings panel.

Properties

Warning Type

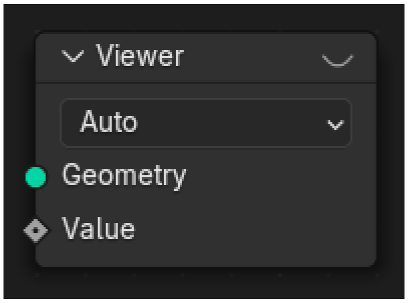
The type of message to display, the type affects the icon displayed.

Outputs

Show

A passthrough of the *Show* input.

Viewer Node



The *Viewer* node allows viewing data from inside a geometry node group in the Spreadsheet Editor and the 3D Viewport.

Any geometry connected can be visualized in the viewport and its attribute values can be read in the spreadsheet.

Note

This node cannot be used in the Tool context—only in the *Modifier* context.

Usage

Activation and Deactivation

Using `Shift-Ctrl-LMB` on any node or socket connects it to the viewer and makes it active. Using the same shortcut on empty space in the node editor makes deactivates the active viewer. When the active viewer is not visible anymore (e.g. another object is selected, or the current node group is exited), it is deactivated. The icon in the viewer node header can also be used to activate and deactivate it.

In the viewport, the *Show Viewer* option can turn off the viewer node visualization completely in order to see the final output of the object's evaluation instead.

Attribute Visualization

When the viewer has a geometry and a separate value input connected, the values can be visualized with a viewport overlays. When possible, the attribute domain is used to visualize the data is determined automatically. Otherwise, the viewer node falls back to the face corner domain on meshes and the point domain on curves. When necessary, the domain can be chosen manually.

The spreadsheet now only shows the “Viewer” column for the domain that is selected in the Viewer node.

Pinning

It can be helpful to pin a specific viewer node in the spreadsheet. When pinned, the spreadsheet still references the viewer node even when it becomes inactive.

Inputs

Geometry
: Geometry that will be displayed in the Spreadsheet.

Value
: Field to be evaluated on the geometry. The type for this value is chosen automatically when the keyboard shortcut to link an output is pressed. However, if the type must be adjusted manually, it is available in the node editor Sidebar.

Properties

Data Type
: The data type used to evaluate the *Value* input, visible in the node sidebar.

Domain
: The attribute domain used to evaluate the *Value* input. The *Auto* option chooses the domain automatically based on the connected nodes.

Outputs

This node has no outputs.

Geometry Nodes

Nodes that can operate on different geometry types (volume, mesh).

- Read
    - ID Node
    - Index Node
    - Named Attribute Node
    - Normal Node
    - Position Node
    - Radius Node
    - Selection Node
    - Active Element Node
- Sample
    - Geometry Proximity Node
    - Index of Nearest
    - Raycast Node
    - Sample Index Node
    - Sample Nearest Node
- Write
    - Set Geometry Name Node
    - Set ID Node
    - Set Position Node
    - Set Selection Node

- Operations
    - Bake Node
    - Bounding Box Node
    - Convex Hull Node
    - Delete Geometry Node
    - Duplicate Elements Node
    - Merge by Distance Node
    - Split To Instances Node
    - Sort Elements Node

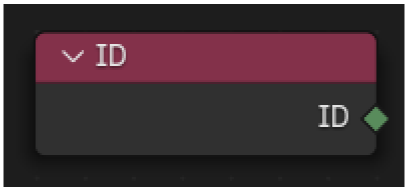
    - Transform Geometry Node
    - Separate Components Node
    - Separate Geometry Node

- Geometry to Instance Node
- Join Geometry Node

Read Geometry Nodes

- ID Node
- Index Node
- Named Attribute Node
- Normal Node
- Position Node
- Radius Node
- Selection Node
- Active Element Node

ID Node



The *ID* node gives an integer value indicating the stable random identifier of each element on the point domain, which is stored in the `id` attribute.

The node to set this data is the Set ID Node node.

Note

Unlike other built-in attributes, the `id` attribute does not always exist. In that case, this node will output the index.

Inputs

This node has no inputs.

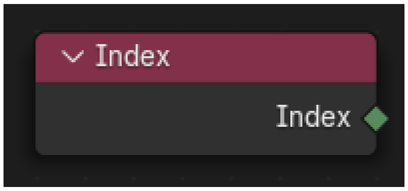
Properties

This node has no properties.

Outputs

ID
: Integer value.

Index Node



The *Index* node gives an integer value indicating the position of each element in the list, starting at zero. This depends on the internal order of the data in the geometry, which is not necessarily visible in the 3D Viewport. However, the index value is visible in the left-most column in the Spreadsheet Editor.

Note

Indices in geometry data are often defined by the internals of complex algorithms that create it. If no inputs change, indices will be the same when the same node tree is executed multiple times. However, they may not be predictable when inputs to nodes that generate geometry or change its topology are adjusted. Additionally, updates to algorithms in newer versions of Blender may change the order of generated elements.

To avoid relying on consistent indices, it is recommended to calculate them locally, or to avoid operations that change topology when they must be consistent over time.

Inputs

This node has no inputs.

Properties

This node has no properties.

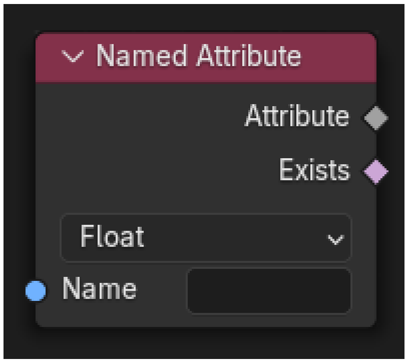
Outputs

Index

Integer value which enumerates each point on the geometry.

Named Attribute Node

The *Named Attribute* node outputs the data of an attribute based on the context of where it is connected (the Field Context).



Inputs

Name

The name of the attribute to read.

Properties

Data Type

The data type used for the retrieved data. Attribute Search can be used to give a basic list of possible attribute names and data types. When a value is chosen from the search menu, the data type is set to automatically choose the data type from the geometry nodes result.

Outputs

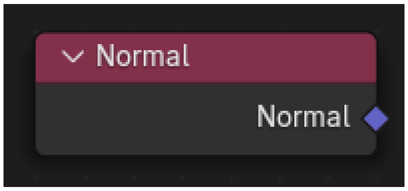
Attribute

The attribute data stored on the geometry.

Exists

True if the attribute accessed by the node is present in the connected context.

Normal Node



The *Normal* node returns a vector for each evaluated point indicating the normal direction. The output can depend on the attribute domain used in the node evaluating the field, but the output is always a normalized unit vector.

Face::	On the face domain, the normal is the “up” direction of the face.
Mesh Vertices::	For mesh vertices, the normal is an average of the surrounding face normals. If the vertex does not have any connected faces, the output is simply the normalized position of that vertex.
Edge::	The normal output for each edge is the average of the edge’s two vertex normals.
Face Corner::	The output for each face corner is the same as the face normal of the corresponding face.
Curve Control Points::	The output of this node when used for curve geometry is the evaluated normal of the curve, which depends on the twist method. The normal vector is always perpendicular to the direction of the curve’s path at every point.

Warning

For NURBS and Bézier spline curves, keep in mind that the value retrieved from this node is the value at every control point, which may not correspond to the visible *evaluated* points. For NURBS splines the difference may be even more pronounced and the result may not be as expected. A Resample Curve Node can be used to create a poly spline, where there is a control point for every evaluated point.

Inputs

This node has no inputs.

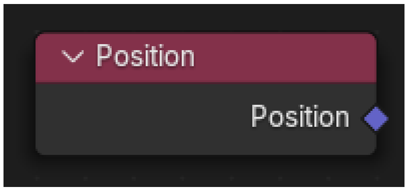
Properties

This node has no properties.

Outputs

Normal
    Vector indicating the normal of each geometry element.

Position Node



The *Position* node outputs a vector of each point of the geometry the node is connected to.

The node can work on geometry domains besides points. In that case, the position data will be automatically interpolated to the new domain. For example, when used as part of the input to the Split Edges Node, the position for each edge will be the average position of the edge's two vertices.

For instances themselves, the output is the origin of each instance. However, if the node is for a geometry node that adjusts data inside instances, the position output of this node will be in the local space of each instance. See the Instance Processing page for more details.

Inputs

This node has no inputs.

Properties

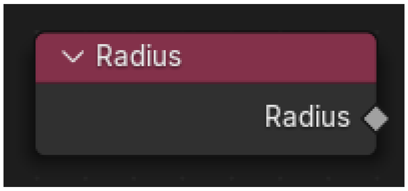
This node has no properties.

Outputs

Position

Vector that indicates the location of each element of the geometry.

Radius Node



The *Radius* node outputs the radius value at each point on the evaluated geometry. For curves, this value is used for things like determining the size of the mesh created in the Curve to Mesh node. For point clouds, the value is used for the display size of the point in the viewport.

Inputs

This node has no inputs.

Properties

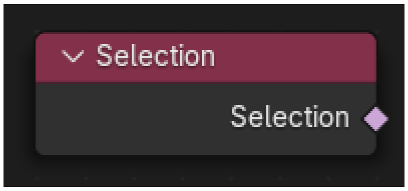
This node has no properties.

Outputs

Radius

Float value indicating radius at each point on the geometry.

Selection Node



The *Selection* node outputs true for geometry that is selected, and false elsewhere.

The corresponding data flow node is the Set Selection Node.

Note

This node can only be used in the Tool context.

Inputs

This node has no inputs.

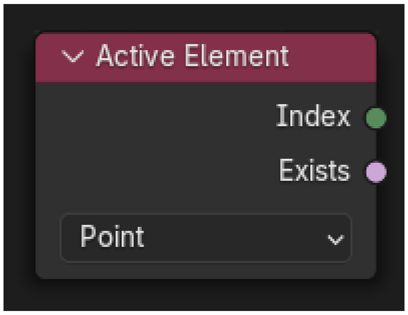
Properties

This node has no properties.

Outputs

Selection
    Boolean field set to true for geometry that is selected in edit mode.

Active Element Node



The *Active Element* node outputs the index of the Active vertex, edge, or face.

Note

This node can only be used in the Tool context.

Inputs

This node has no inputs.

Properties

Domain
  Which domain to return the index of.

Outputs

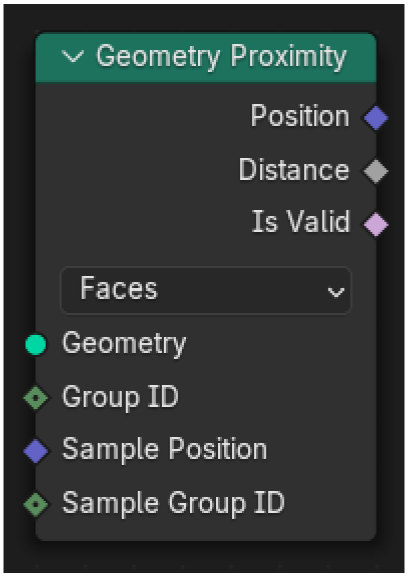
Index
  Index of the active element in the specified domain.

Exists
  True if an active element exists in the mesh, false otherwise

Sample Geometry Nodes

- Geometry Proximity Node
- Index of Nearest
- Raycast Node
- Sample Index Node
- Sample Nearest Node

Geometry Proximity Node


Geometry Proximity
Position
Distance
Is Valid
Faces
Geometry
Group ID
Sample Position
Sample Group ID

The *Geometry Proximity* node computes the closest location on the target geometry.

Tip

The Map Range Node is often helpful to use with the distance output of this node to create a falloff with a maximum distance.

Inputs

Geometry
: Standard geometry input.

Group ID
: Splits the elements of the input geometry into groups which can be sampled individually.

Sample Position

The given position to calculate the closest location on the target.

Sample Group ID

Determines in which group the closest nearest element is detected.

Properties

Target Element

Faces::	Calculate the closest point anywhere on the faces of the target's mesh geometry.
Edges::	Calculate the closest point anywhere on the edges of the target's mesh geometry.
Points::	Calculate the closest point or vertex on the target geometry. This mode is usually the fastest. This mode works for both point cloud and mesh geometry, the other modes only work for meshes.

Outputs

Position

Closest location on the surface of the target mesh, or the closest point in the target point cloud in *Points* mode.

Distance

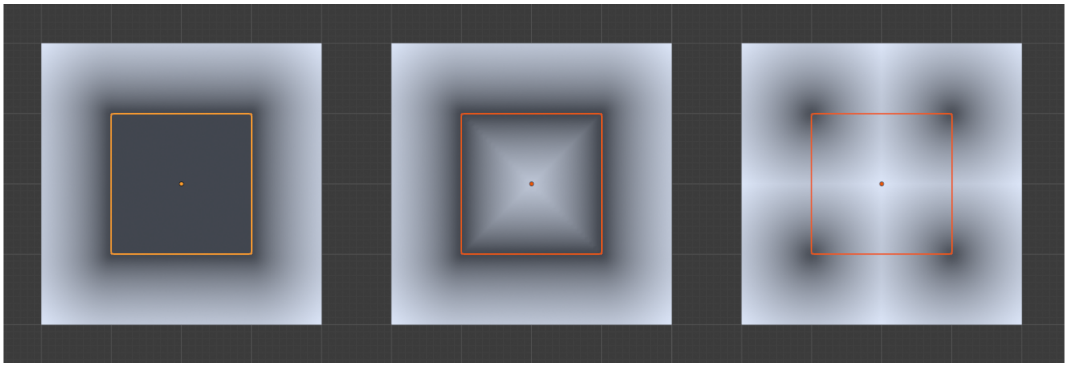
Distance (as floating-point value) from the source position to the closest location in the target.

Is Valid

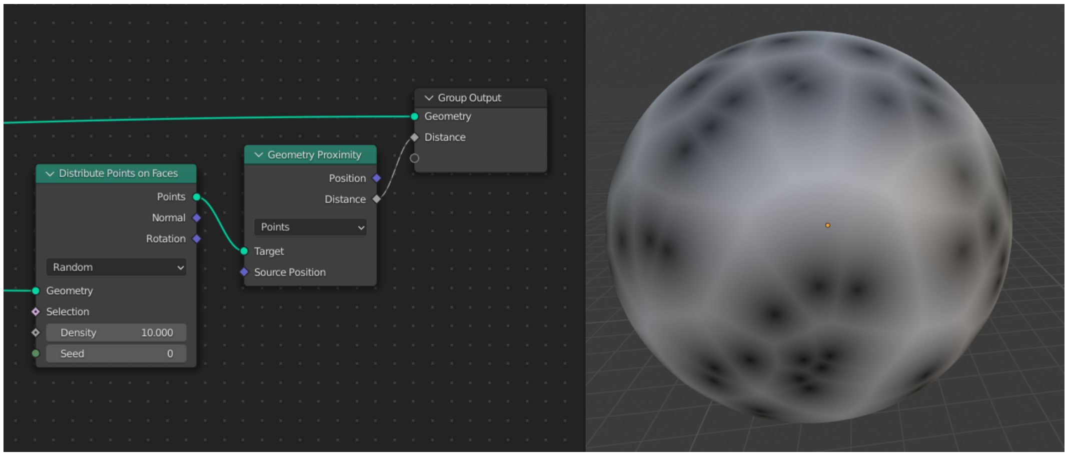
Whether the sampling was successful. It can fail when the sampled group is empty.

Examples

The different modes of the node: faces, edges and points. In this example the Geometry Nodes modifier is added on the target plane. Note that the larger plane is subdivided and the smaller plane is not.

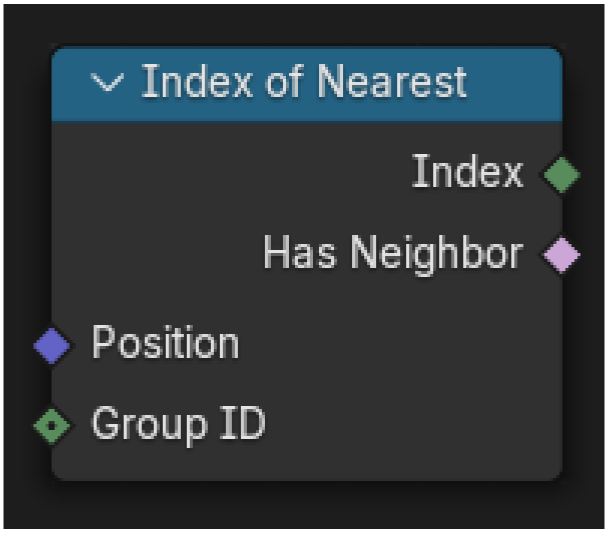


The three target element modes: faces, edges, and points.


Group Output
Geometry
Distance
Geometry Proximity
Position
Distance
Points
Target
Source Position
Distribute Points on Faces
Points
Normal
Rotation
Random
Geometry
Selection
Density 10.000
Seed 0

Points distributed on a sphere used as a target for a distance used in a shader.

Index of Nearest



The **Index of Nearest** node is a way to find other close elements in the same geometry. If needed you can use Group ID to determine the group of neighbors to be analyzed together.

This is an alternative to the Sample Nearest Node node. The main difference is that this node does not require a geometry input, because the geometry from the field context is used.

Tip

This is often combined with the Evaluate at Index Node or the Sample Index Node node.

Inputs

Position
: The position for each element to search. By default, this is the same as if the Position Node was connected.

Group ID
: ID to group elements together.

Outputs

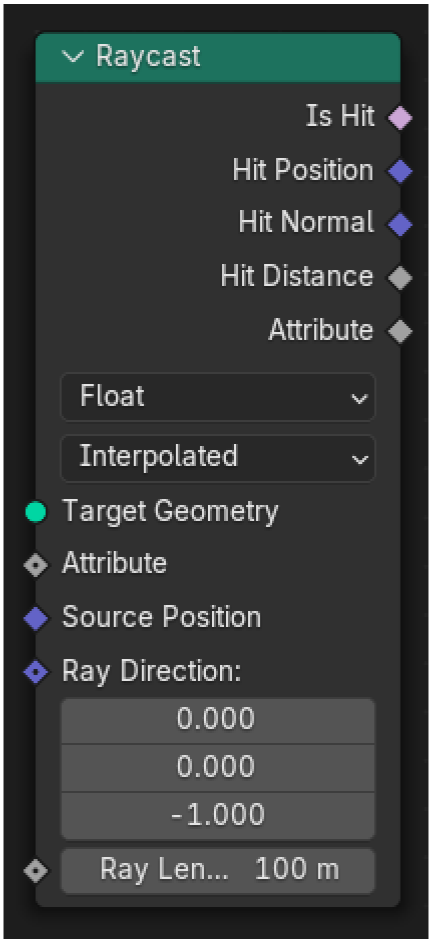
Index

The index of the closest element in the same geometry component.

Has Neighbor

This is true when the group of the element has at least two elements. This is only relevant when using *Group ID*.

Raycast Node


Raycast
Is Hit
Hit Position
Hit Normal
Hit Distance
Attribute
Float
Interpolated
Target Geometry
Attribute
Source Position
Ray Direction:
0.000
0.000
-1.000
Ray Len... 100 m

The *Raycast* node intersects rays from one geometry onto another. The source geometry is defined by the context of the node that the *Raycast* node is connected to. Each ray computes hit points on the target mesh and outputs normals, distances and any surface attribute specified.

Inputs

Target Geometry
 Geometry that rays are tested against.

Attribute

An optional field input evaluated on the *Target Geometry* that will be interpolated at the hit points. The resulting values are outputted with the *Attribute* output.

Source Position

The position from where to start each ray. By default, this is the same as if the Position Node was connected.

Ray Direction

Direction of each ray from the starting position. The field is evaluated on the geometry from the context of the field evaluation, not the *Target Geometry*.

Ray Length

Maximum distance a ray can travel before being considered “no hit”.

Properties

Mapping

How attributes of the target mesh are mapped to the attribute values on the result geometry.

Interpolated::	Vertex and corner attributes are interpolated smoothly, with a bilinear function.
Nearest::	Choose the value of the closest vertex without interpolating.

Outputs

Is Hit

Boolean output that is true for each ray which has hit the *Target Geometry*.

Hit Position

The location of the intersection point with the target mesh.

Hit Normal

The surface Normal vector at the hit location.

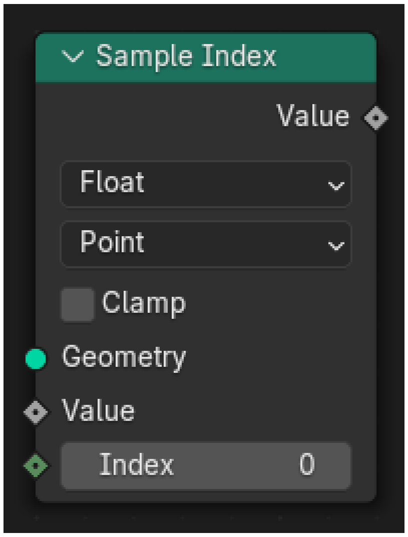
Hit Distance

The distance from the *Source Position* to the *Hit Position*. If the ray does not hit, the *Ray Length* is returned.

Attribute

Interpolated values of the *Attribute* input sampled at the *Hit Position*.

Sample Index Node



The *Sample Index* node retrieves values from a source geometry at a specific index.

Tip

If the *Geometry* used for the input is the same as the geometry from the field context, this node is equivalent to the Evaluate at Index Node. Using that node is usually preferable since avoiding the geometry socket makes the whole setup easier to use in other situations and share.

Tip

Different components can have same attribute domain (Points). This node simply uses first component that not empty for such domain, checked in the order of: Mesh, Point Cloud, Curve. The Separate Components Node can be used to sample directly from a specific component.

Inputs

Geometry
    The geometry to retrieve the attribute from.

Value
    A field to evaluate on the source *Geometry*. The values are then retrieved from specific indices for the output.

Index
    Which index to use when retrieving the data from the input *Value* field. Any index can be connected, resulting in a “shuffling” of the values.

Properties

Data Type
    The data type to use for the retrieved values.

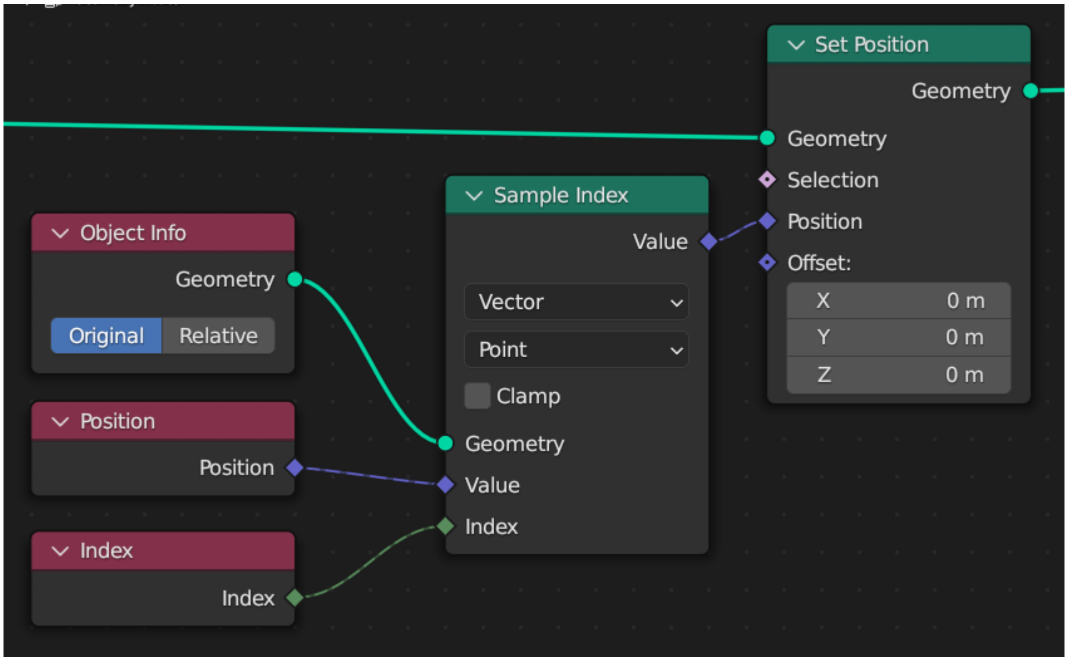
Domain
    The attribute domain that the attribute is transferred from, or in other words, the domain used to evaluate the *Attribute* input. For example, it is possible to transfer data from the faces of one geometry to the points of another.

Clamp
    Clamp the indices to the size of the attribute domain instead of outputting a default value for invalid indices.

Outputs

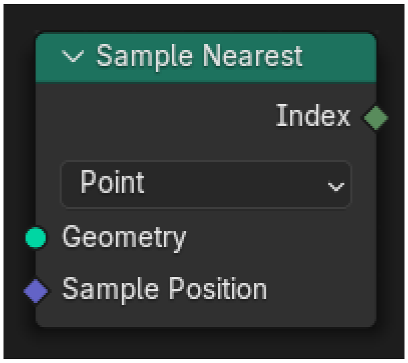
Value
    The data retrieved from the source *Geometry* input.

Examples


Set Position
Geometry
Geometry
Selection
Position
Offset:
X 0 m
Y 0 m
Z 0 m
Sample Index
Value
Vector
Point
Clamp
Geometry
Value
Index
Object Info
Geometry
Original
Relative
Position
Position
Index
Index

Here the node is used to copy the positions of one object to another. This recreates the behavior of the *Transfer Attribute* node from Blender versions before 3.4. This works best when their geometries have the same number of points and the same Topology.

Sample Nearest Node



The *Sample Nearest* node retrieves the index of the geometry element in its input geometry that is closest to the input position.

This node is similar to the Geometry Proximity Node, but it outputs the index of the closest element instead of its distance from the current location.

Tip

If you want to find nearest to each point in same geometry, its better to use the Index of Nearest node.

Inputs

Geometry
    The geometry to sample.

Note

This node only supports point cloud and mesh inputs.

Sample Position

The position to start from when finding the closest location on the target geometry. By default, this is the same as if the Position Node was connected.

Properties

Domain

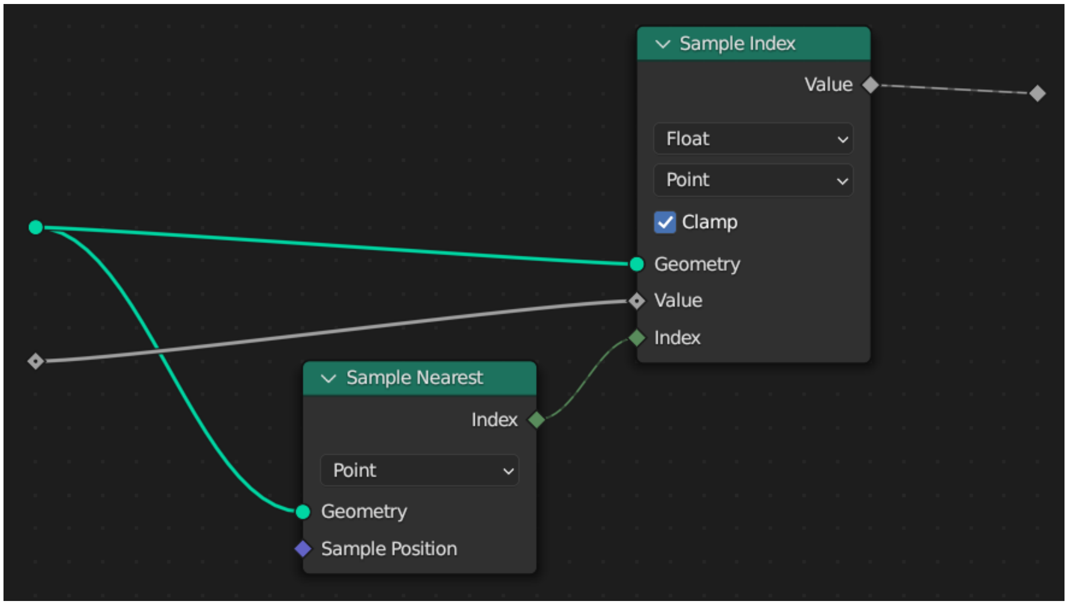
The attribute domain to consider the distance from.

Outputs

Index

The index of the closest geometry element of the chosen domain.

Examples

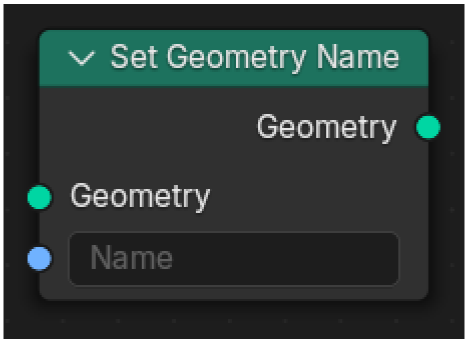

Sample Index
Value
Float
Point
☑ Clamp
Geometry
Value
Index
Sample Nearest
Index
Point
Geometry
Sample Position

Combining this node with the Sample Index Node gives a setup that can retrieve the closest attribute value from another geometry. This is the same behavior as the *Transfer Attribute* node in versions of Blender before 3.4.

Write Geometry Nodes

- Set Geometry Name Node
- Set ID Node
- Set Position Node
- Set Selection Node

Set Geometry Name Node



The *Set Geometry Name* node stores a custom name on the geometry, overriding the name which might come from the Object Info Node or a Grease Pencil to Curves Node. The name is displayed in the spreadsheet and can helful for debugging purposes.

Inputs

Geometry
    Standard geometry input.

Name
    The new name for the geometry.

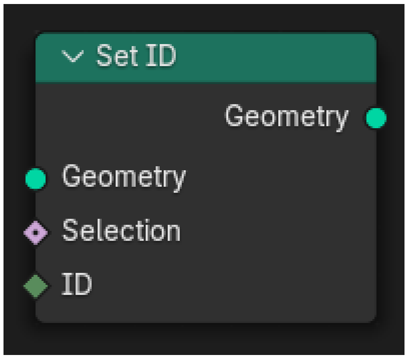
Properties

This node has no properties.

Outputs

Geometry
    Standard geometry output.

Set ID Node



The *Set ID* node fills the `id` attribute on the input geometry. If the attribute does not exist yet, it will be created with a default value of zero. The ID is also created by the Distribute Points on Faces, and it is used in the Random Value Node and other nodes if it exists.

The input node for this data is the ID Node.

Inputs

Geometry
: Standard geometry input.

Selection
: Whether or not to change the value on each point or instance. True values mean the value will be changed, false values mean the value will remain the same.

ID
: The value for each element. By default, this input uses the index, which is useful when stable IDs are desired when deleting a dynamic number of instances.

Properties

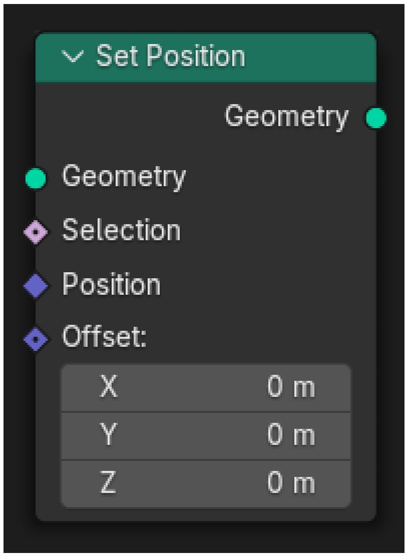
This node has no properties.

Outputs

Geometry

Standard geometry output.

Set Position Node



The *Set Position* node controls the location of each point, the same way as controlling the `position` attribute. If the input geometry contains instances, this node will affect the location of the origin of each instance.

The input node for this data is the Position Node.

Inputs

Geometry
: Standard geometry input.

Selection
: Whether or not to change the position of each point or instance. True values mean the position will be changed, false values mean it will remain the same.

Position
: The new position for selected elements. By default, this is the same as if the Position Node was connected, meaning the node will do nothing.

Offset
: An optional translation for each point. This is evaluated at the same time as the *Position* input, meaning that fields evaluated for it will not reflect the changed position.

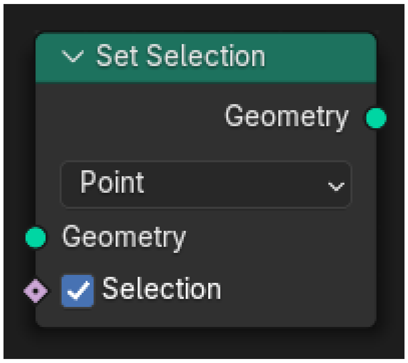
Properties

This node has no properties.

Outputs

Geometry
    Standard geometry output.

Set Selection Node



The *Set Selection* node controls which geometry is selected.

The input node for this data is the Selection Node.

Note

This node can only be used in the Tool context.

Inputs

Geometry
: Standard geometry input.

Selection
: Boolean field for specifying which elements should be selected in the output geometry. Elements for which this field evaluates to false are implicitly de-selected.

Properties

Domain
: Which domain to set the selection on.

Outputs

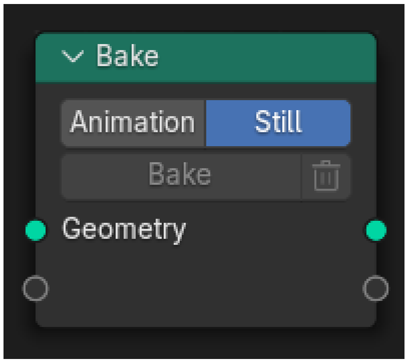
Geometry
: Standard geometry output.

Geometry Operation Nodes

- Bake Node
- Bounding Box Node
- Convex Hull Node
- Delete Geometry Node
- Duplicate Elements Node
- Merge by Distance Node
- Split To Instances Node
- Sort Elements Node
- Transform Geometry Node

- Separate Components Node
- Separate Geometry Node

Bake Node



The *Bake* node allows saving and loading intermediate geometries. This node bakes parts of the node tree for better performance.

The data format used to store geometry data is not considered to be an import/export format. Volume objects, however, are saved using the OpenVDB file format which can be used interoperably.

Important

- It's not guaranteed that data written with one Blender version can be read by another Blender version.

Inputs

Geometry
Standard geometry input, which is used as the default bake item. More bake items can be added by dragging sockets into the blank socket or in the *Bake Items* panel. Items can be renamed by `Ctrl-LMB` on the socket name or in the nodes *Properties* panel.

Bake Items

Reference

Editor::	Geometry Node Editor
Panel::	*Sidebar* ▸ *Node* ▸ *Bake Items*

The *Bake Items* panel is used to manage the input sockets of the node thus also managing what data is baked.

Bake Items List
: Used to manage the inputs and outputs of the bake node. Items can be added, removed, renamed, and sorted.

Socket Type
: The data (Sockets) of the input/output.

Attribute Domain
: The attribute domain used to evaluate the input field on.

Is Attribute
: Bake item is an attribute stored on a geometry.

Properties

Note

Some properties can only be edited in the Properties panel (*Sidebar* ‣ *Node* ‣ *Properties*).

Bake Mode
: The *Bake* node can calculate the geometry of a single frame or an animation.

Animation::	Bakes the geometry data for multiple frames. By default the scene frame range is used, however, a *Custom Range* can also be defined.
Still::	Bakes the geometry data of the current frame.

Bake
: Preforms all necessary geometry calculations and saves the data on disk or packed into the .blend file.

Pack/Unpack

Packs or unpacks already baked data. When unpacking, there are different options to control where the data will be stored. The options are the same as for the Unpack Resources operator.

Delete (Trash Icon)

Deletes the bake data.

Bake Target

Specifies where the baked data should be stored.

Inherit from Modifier::	The setting is copied from the modifier that contains the bake node. This is the default.
Packed::	The baked data is packed into the .blend file. So no separate file is necessary.
Disk::	The baked data is stored in a separate directory on disk.

Custom Path

Specify a path where the baked data should be stored manually.

Bake Path

Location on disk where the baked data is stored. Note, this path is also used for simulation zones.

Custom Range *Animation*

Override the simulation frame range from the scene.

Start, End

The start and end frame numbers for the custom range.

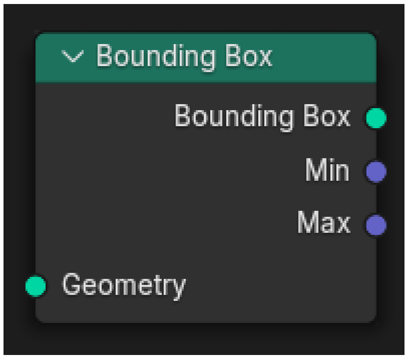
Outputs

For each input, the same output is added to act as a pass through.

Geometry

Standard geometry output, which is used as the default bake item. More bake items can be added by dragging sockets into the blank socket or in the *Bake Items* panel. Items can be renamed by `Ctrl-LMB` on the socket name or in the nodes *Properties* panel.

Bounding Box Node



The *Bounding Box* node creates a box mesh with the minimum volume that encapsulates the geometry of the input. The node also can output the vector positions of the bounding dimensions.

The mesh output and the *Min* and *Max* outputs do not take instances into account. Instead, for instanced geometry, a bounding box is computed for each instance rather than the whole geometry. To compute the bounding box including the instances, a Realize Instances Node can be used.

Inputs

Geometry
: Standard geometry input.

Properties

This node has no properties.

Outputs

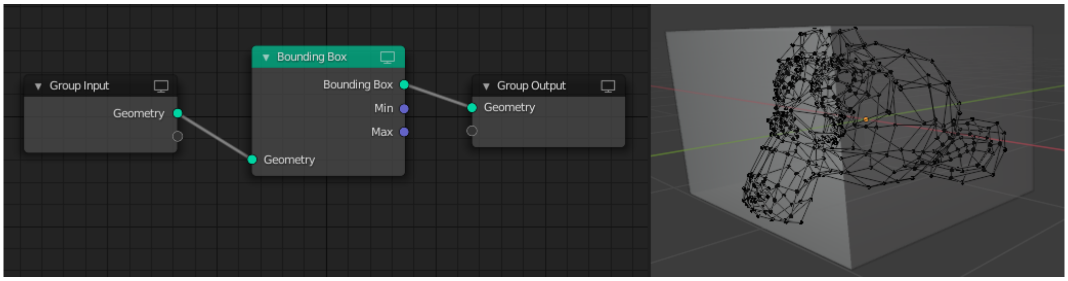
Bounding Box
: The resulting box that encapsulate the input geometry.

Min
: The coordinates corresponding to the box's -X, -Y, -Z position values, i.e. how far the box extends in each of the negative axes directions.

Max

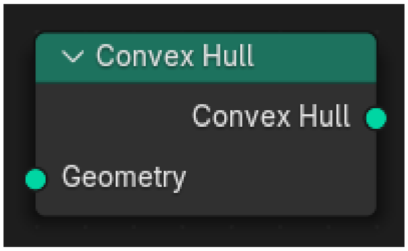
The coordinates corresponding to the box’s +X, +Y, +Z position values, i.e. how far the box extends in each of the positive axes directions.

Example


Group Input
Geometry
Bounding Box
Bounding Box
Min
Max
Geometry
Group Output
Geometry

Bounding Box node used to create a box that encapsulates the geometry of the monkey mesh.

Convex Hull Node



The *Convex Hull* node outputs a convex mesh that is enclosing all points in the input geometry.

Note

When the node is used on a geometry with instances, the algorithm will run once per instance, resulting in many convex hull meshes in the instance geometries. The *Realize Instances* node can be used to get a convex hull of an entire geometry.

Important

Volumes are not supported by this node, and attributes are not automatically transferred to the result.

Inputs

Geometry
Standard geometry input.

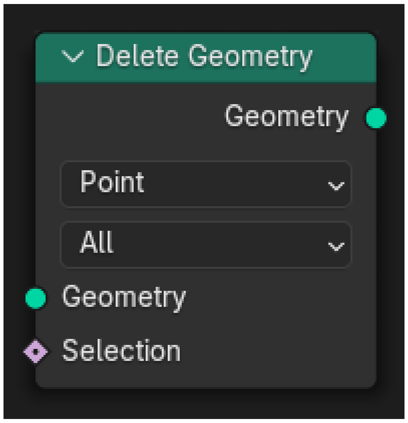
Properties

This node has no properties.

Outputs

Convex Hull
    Mesh that encloses all points in the input.

Delete Geometry Node



The *Delete Geometry* node removes the selected part of a geometry. It behaves similarly to the Delete tool in Edit Mode. The type of elements to be deleted can be specified with the domain and mode properties.

Inputs

Geometry
Standard geometry input.

Selection
Boolean field that is true for parts of the geometry to be deleted.

Properties

Domain
The domain on which the selection field is evaluated.

Point::	The selection is on the points, control points, and vertices of the geometry.
Edge::	The selection is on the edges of the mesh component. The other components are not modified.

Face::	The selection is on the faces of the mesh component. The other components are not modified.
Spline::	The selection is on the splines in the curve component. For each spline, it will either be deleted entirely or not at all. The other components are not modified.
Instance::	The selection is on the top-level instances, and the entire instance is removed if it is selected. If the input also contains realized geometry, that will be unaffected.

Mode

The type of elements to be affected. This only applies to the mesh component.

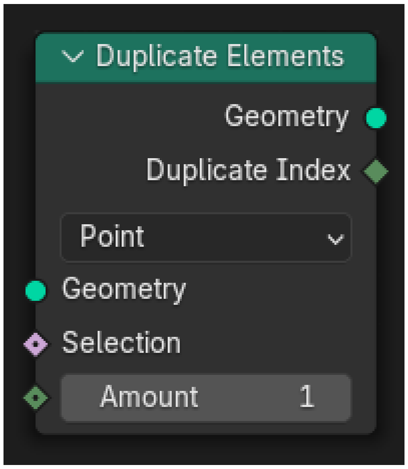
All::	Vertices, edges, and faces in the selection will be deleted.
Only Edges & Faces::	Vertices won't be deleted, even if they are in the selection.
Only Faces::	Only faces in the selection will be deleted.

Output

Geometry

Standard geometry output.

Duplicate Elements Node


Duplicate Elements
Geometry
Duplicate Index
Point
Geometry
Selection
Amount 1

The *Duplicate Elements* node creates a new geometry with the specified elements from the input duplicated an arbitrary number of times. The positions of elements are not changed, so all of the duplicates will be at the exact same location.

Inputs

Geometry
: Standard geometry input.

Selection
: Boolean field that is true for parts of the geometry to be deleted.

Amount
: Field indicating how many times each input element should be duplicated. If the value is zero for an element, it will not be included in the output at all.

Properties

Domain

The type of geometry element to duplicate

Point::	Duplicate the points of meshes, curves, or point clouds. Any other elements will not be included in the output.
Edge::	Duplicate mesh edges. Faces will not be included in the output.
Faces::	Duplicate mesh faces. Each duplicated face will be separate, in other words they will not share edges with other faces.
Spline::	Individual curves from the input curves component will be duplicated.
Instances::	Input top-level instances will be duplicated.

Output

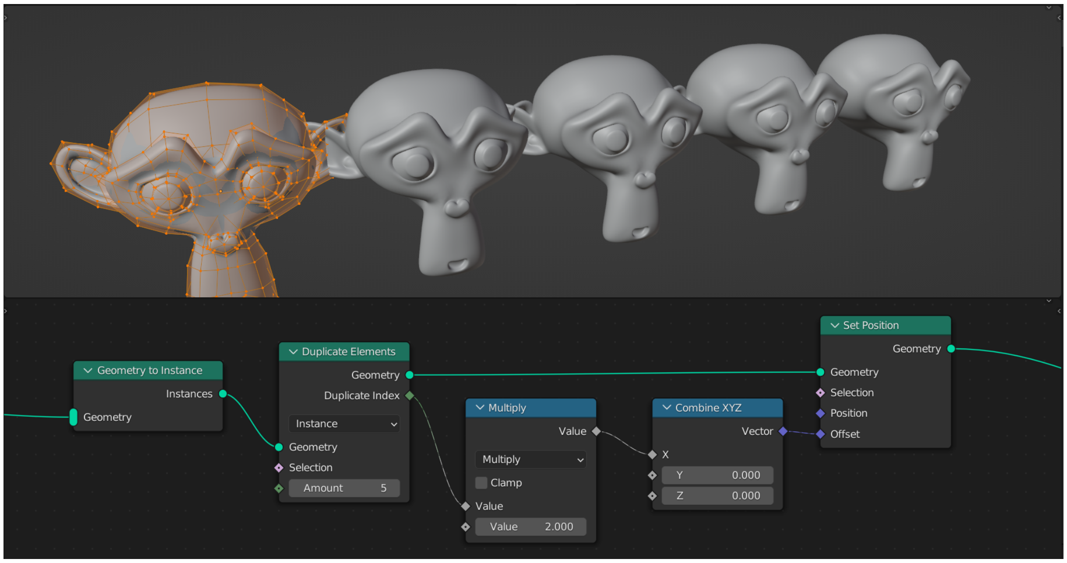
Geometry

Standard geometry output.

Duplicate Index

An attribute field with a value for every output element describing which duplicate of the corresponding input. The value for every input element will start at 0 and increase to `Amount - 1`.

Examples


Geometry to Instance
Instances
Geometry
Duplicate Elements
Geometry
Duplicate Index
Instance
Geometry
Selection
Multiply
Value
Multiply
Clamp
Value
Combine XYZ
Vector
X
Set Position
Geometry
Geometry
Selection
Position
Offset

Amount 5

Value 2.000

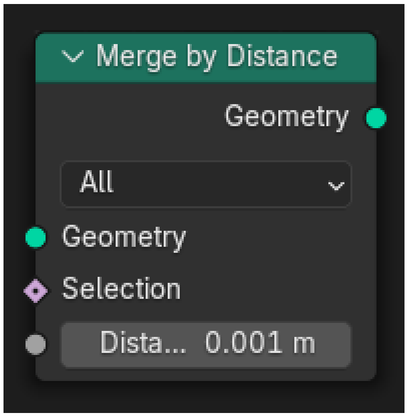
Y 0.000

Z 0.000

Combined with the Geometry to Instance Node, this can be used to create a basic efficient "Array" operation. This should be more efficient because the duplicates are instances.

The "Duplicate Index" is used to move each instance in the result a different amount.

Merge by Distance Node



The *Merge by Distance* node merges selected mesh vertices or point cloud points within a given distance, merging surrounding geometry where necessary. This operation is similar to the Merge by Distance operator or the Weld Modifier.

Inputs

Geometry
: Standard geometry input.

Selection
: Boolean field that is true for parts of the geometry to be deleted. Unselected points will be completely unused for the operation– they will not be merged into other points, and no points will merge into them either.

> **Tip**
>
> When possible, using the selection input can be a simple way to speed up the node, since searching for nearby points is a relatively expensive operation that gets even more expensive when more points are involved.

Distance
: The distance to use for searching for nearby points.

Properties

Mode

Method for choosing which vertices are merged.

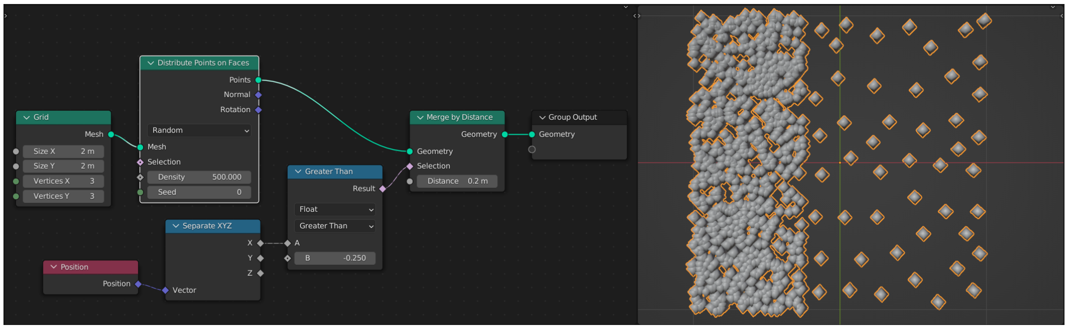
All::	Merge includes all geometry including loose parts.
Connected::	Merge only includes attached geometry i.e. the modifier will not merge loose parts together.

Output

Geometry

Standard geometry output.

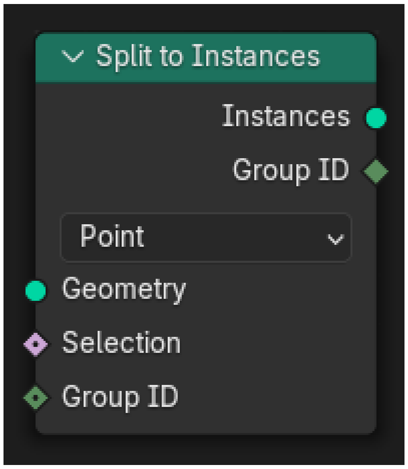
Examples


Grid
Mesh
Size X 2 m
Size Y 2 m
Vertices X 3
Vertices Y 3
Distribute Points on Faces
Points
Normal
Rotation
Random
Mesh
Selection
Density 500.000
Seed 0
Position
Position
Separate XYZ
X
Y
Z
Vector
Greater Than
Result
Float
Greater Than
A
B -0.250
Merge by Distance
Geometry
Geometry
Selection
Distance 0.2 m
Group Output
Geometry

Using the selection input to only merge some of the points in a point cloud.

Split To Instances Node

Splits a selection of geometry elements (such as faces) into groups, then turns each group into an instance.


Split to Instances
Instances
Group ID
Point
Geometry
Selection
Group ID

Inputs

Geometry

Standard geometry input.

Selection

Boolean field indicating which geometry elements to include.

Group ID

Integer field indicating which group each element belongs to. Elements with the same ID will be moved into the same output instance.

Properties

Domain

The type of geometry to extract and split. This is also the domain on which the *Selection* and *Group ID* fields are evaluated.

Point::	Points, spline control points, and vertices.
Edge::	Mesh edges.
Face::	Mesh faces.

Spline::	Curve splines.
Instance::	Top-level instances. Realized instances are ignored.
Layer::	Grease Pencil layers.

Note

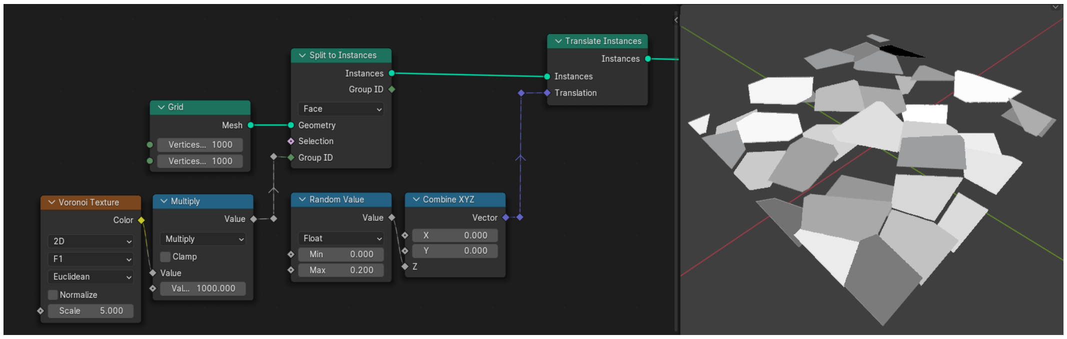
Geometry that doesn’t match the selected domain will be removed. For example, if you choose *Edge*, any faces, splines, and instances in the input geometry will be lost.

Output

Instances
: The instances containing the grouped geometry elements.

Group ID
: Group ID of each instance.

Examples

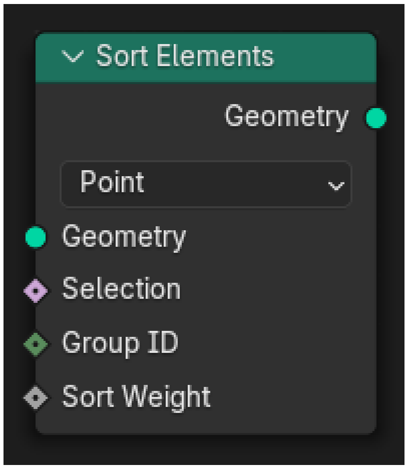

Translate Instances
Instances
Instances
Translation
Split to Instances
Instances
Group ID
Face
Geometry
Selection
Group ID
Grid
Mesh
Vertices... 1000
Vertices... 1000
Voronoi Texture
Color
2D
F1
Euclidean
Normalize
Scale 5.000
Multiply
Value
Multiply
Clamp
Value
Val... 1000.000
Random Value
Value
Float
Min 0.000
Max 0.200
Combine XYZ
Vector
X 0.000
Y 0.000
Z

In the example above, we start with a grid of 1000x1000 square faces serving as “pixels.” Then, we group these faces into patches by assigning them a group ID sampled from a Voronoi texture, and move each resulting instance by a random amount along the Z axis.

Note that, because the texture outputs floating point values between 0 and 1 while the group ID is an integer, all the values would be rounded to 0 or 1 and we would only get two groups. To get more variation, we multiply the texture value by 1000.

Sort Elements Node

The *Sort Elements* node rearranges geometry elements by changing their indices.


Sort Elements
Geometry
Point
Geometry
Selection
Group ID
Sort Weight

Inputs

Geometry

Standard geometry input.

Selection

The selection of elements to sort, if left blank, all elements are sorted. Non selected elements will be keep their current indices.

Group ID

Elements with the same group ID are sorted together. If this is not a field, the node has no affect.

Sort Weight

The sorted values used to do the reordering. If this is not a field, the node has no affect.

Properties

Domain

The domain on which the selection and group ID fields are evaluated.

Point::	The fields are evaluated on points, control points, and vertices.
Edge::	The fields are evaluated on the edges of the mesh component.
Faces::	The fields are evaluated on the faces of the mesh component.
Spline::	The fields are evaluated on the splines in the curve component.
Instance::	The fields are evaluated on the top-level instances. Realized instances are ignored.

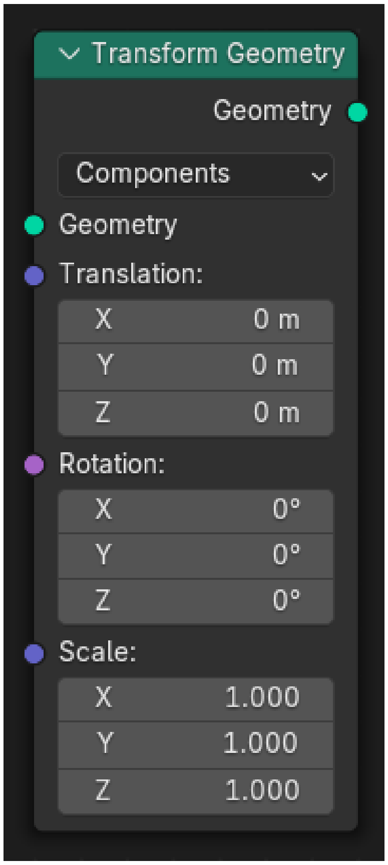
Outputs

Geometry

Standard geometry output.

Transform Geometry Node

The *Transform Geometry Node* allows you to move, rotate or scale the geometry. The transformation is applied to the entire geometry, and not per element. The Set Position Node is used for moving individual points of a geometry. For transforming instances individually, the instance translate, rotate, or scale nodes can be used.



Translation:
X 0 m
Y 0 m
Z 0 m
Rotation:
X 0°
Y 0°
Z 0°
Scale:
X 1.000
Y 1.000
Z 1.000

Inputs

Geometry
: Standard geometry input.

Translation
: Translation of the entire geometry in the local space of the modified object.

Rotation
: Euler rotation in the local space of the modified object.

Scale

Scale for the geometry in the local space of the modified object.

Transform
A Transformation Matrix, available when using *Matrix* mode.

Properties

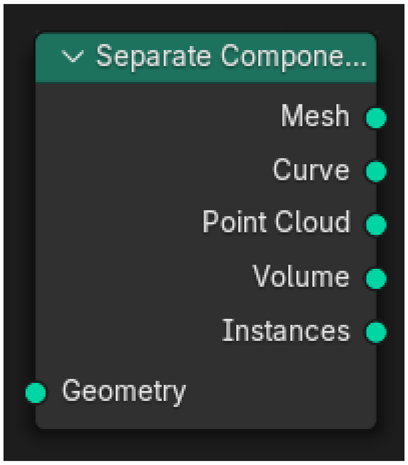
Mode
How the transformation is specified.

Components::	Provide separate inputs for location, rotation and scale.
Matrix::	Use a transformation matrix.

Output

Geometry
Standard geometry output.

Separate Components Node


Separate Compone...
Mesh
Curve
Point Cloud
Volume
Instances
Geometry

The *Separate Components* node splits a geometry into a separate output for each type of data in the geometry.

Inputs

Geometry
: Standard geometry input.

Properties

This node has no properties.

Outputs

Mesh
: Mesh component of the input geometry.

Curve
: Curve component of the input geometry.

Point Cloud

Point cloud component of the input geometry.

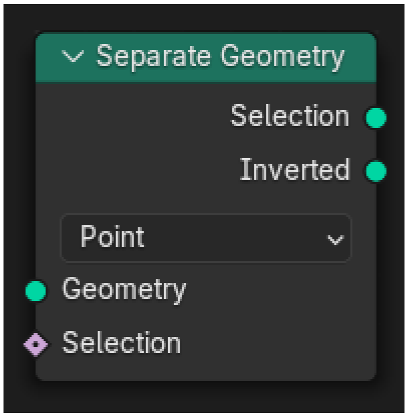
Volume

Volume component of the input geometry.

Instances

Instances component of the input geometry. Even if the instances contain geometry data with one of the other types, all instances will be added to this output. A Realize Instances Node can be added to move the data from geometry instances to their corresponding outputs.

Separate Geometry Node



The *Separate Geometry* node produces two geometry outputs. Based on the *Selection* input, the input geometry is split between the two outputs.

Tip

This node can be combined with the Compare Node for a more precise control of which parts are separated to a given output geometry.

Inputs

Geometry
Standard Geometry input.

Selection
Boolean field used to calculate which output each part of the geometry will go to. Parts in the selection will move to the *Selection* output. Parts not in the selection will move to the *Inverted* output.

Properties

Domain
The domain on which the selection field is evaluated.

Point::	The selection is on the points, control points, and vertices of the geometry.
Edge::	The selection is on the edges of the mesh component. The other components are not modified.

Faces::	The selection is on the faces of the mesh component. The other components are not modified.
Spline::	The selection is on the splines in the curve component. For each spline, it is either entirely in the selection or not at all. The other components are not modified.

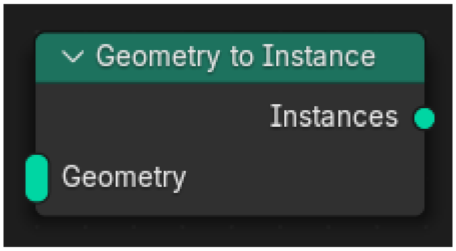
> **Note**
>
> When selecting a domain that doesn't modify all components, the unmodified components will appear in both outputs.

Outputs

Selection
: The parts of the geometry in the selection.

Inverted
: The parts of the geometry not in the selection.

Geometry to Instance Node



The *Geometry to Instance* node turns every connected input geometry into an instance. Visually, the node has a similar result as the Join Geometry Node, but it outputs the result as separate instances instead. The geometry data itself isn't actually joined.

The node can be used in combination with the *Pick Instances* option in the Instance on Points Node, as a way to pick between geometry generated in the node tree (as opposed to picking from separate instances from the Collection Info Node, for example).

Tip

This node can be much faster than the join geometry node when the inputs are large geometries. This is because the join geometry node must actually create a larger mesh, or a larger curve. Even though the operation is simple, just creating a large mesh can have a significant cost. This node can be better, because instead of merging large geometries, it just groups them together as instances.

Inputs

Geometry
: Geometry that will be joined. Multiple inputs are allowed. When the node is muted, only the first link will be passed through.

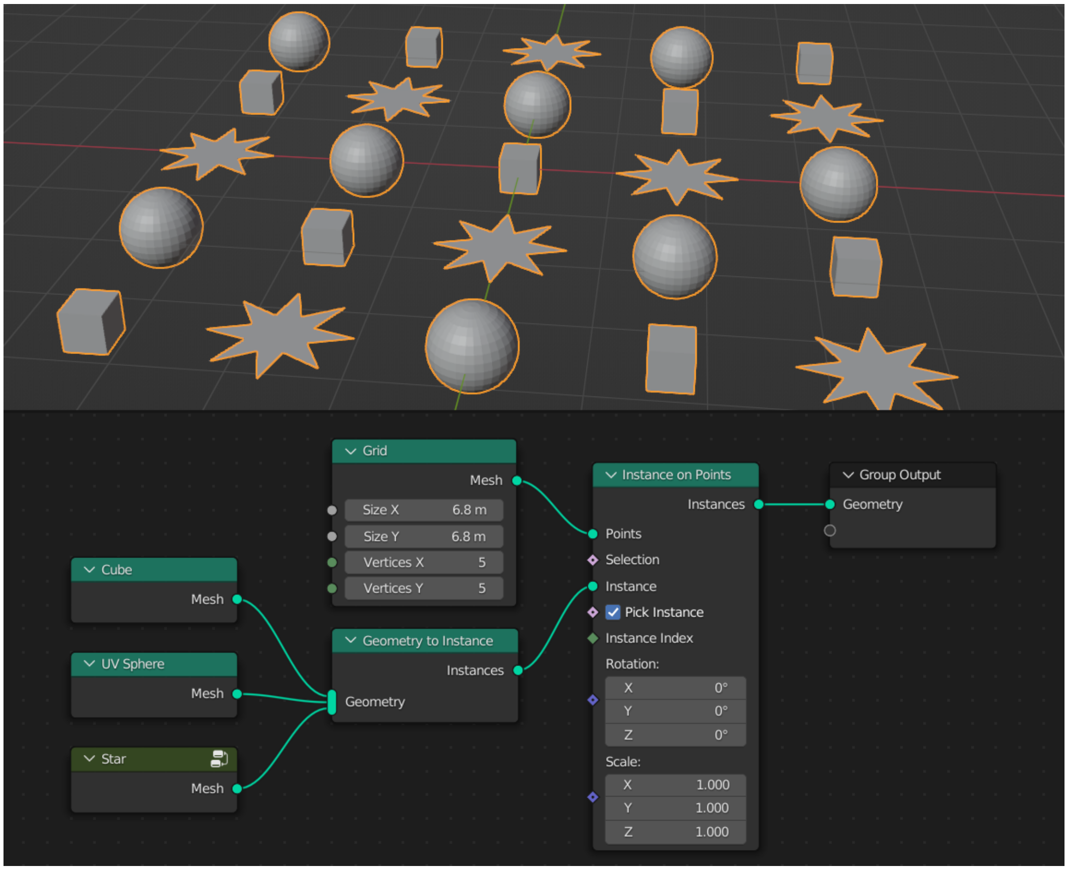
Properties

This node has no properties.

Output

Geometry
  Standard geometry output.

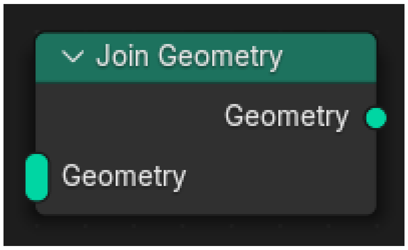
Examples


Grid
Mesh
Size X 6.8 m
Size Y 6.8 m
Vertices X 5
Vertices Y 5
Instance on Points
Instances
Points
Selection
Instance
Pick Instance
Instance Index
Rotation:
X 0°
Y 0°
Z 0°
Scale:
X 1.000
Y 1.000
Z 1.000
Group Output
Geometry
Cube
Mesh
UV Sphere
Mesh
Star
Mesh
Geometry to Instance
Instances
Geometry

The node used in combination with the [Instance on Points Node]() to choose between multiple primitives for instancing.

Join Geometry Node

The *Join Geometry* node merges separately generated geometries into a single one. If the geometry inputs contain different types of data, the output will also contain different data types.



Note

The node cannot handle the case when more than one geometry input has a volume component.

Materials

When multiple mesh inputs contain different materials, the material slots from each mesh geometry are merged so that the output mesh will contain all the input materials.

Attributes

When merging attributes from multiple geometry inputs, the highest complexity data type is chosen for the output attribute. In other words, if a `weight` attribute has a Boolean type on one geometry input and a vector data type on another geometry, the `weight` attribute on the output geometry will have a vector data type. The same heuristic is used for attribute domains, the domain with the most information will be used for the output.

Warning

Like other geometry nodes, this node always outputs generic typed attributes. So instead of a Vertex Group attribute, it will create a “Float” attribute on the result, and it will create a generic 2D vector attribute instead of a special “UV Map” attribute. Some other areas of Blender don’t properly handle generic attributes in version 3.0.

Custom face corner normals are also not transferred currently.

Inputs

Geometry
Geometry that will be joined. Multiple inputs are allowed. When the node is muted, only the first link will be passed through.

Properties

This node has no properties.

Output

Geometry
Standard geometry output.

Curve Nodes

Nodes that only operate on curves.

Note

If any modifiers precede the Geometry Nodes modifier, the curve will be seen as a mesh internally, and the Curve Nodes won't work. To get around this, first run the geometry through a Mesh to Curve Node.

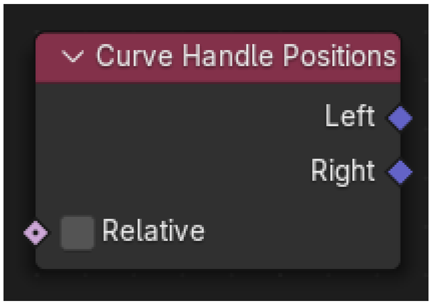
- Read
    - Curve Handle Positions Node
    - Curve Length Node
    - Curve Tangent Node
    - Curve Tilt Node
    - Endpoint Selection Node
    - Handle Type Selection Node
    - Is Spline Cyclic Node
    - Spline Length Node
    - Spline Parameter Node
    - Spline Resolution Node
- Sample
    - Sample Curve Node
- Write
    - Set Curve Normal Node
    - Set Curve Radius Node
    - Set Curve Tilt Node
    - Set Handle Positions Node
    - Set Handle Type Node
    - Set Spline Cyclic Node
    - Set Spline Resolution Node
    - Set Spline Type Node

- Operations
    - Curve to Mesh Node
    - Curve to Points Node
    - Curves to Grease Pencil Node
    - Deform Curves on Surface Node
    - Fill Curve Node
    - Fillet Curve Node
    - Grease Pencil to Curves Node
    - Interpolate Curves Node
    - Merge Layers Node
    - Resample Curve Node
    - Reverse Curve Node
    - Subdivide Curve Node
    - Trim Curve Node
- Primitives
    - Arc Node
    - Bézier Segment Node
    - Curve Circle Node
    - Curve Line Node
    - Curve Spiral Node
    - Quadratic Bézier Node
    - Quadrilateral Node
    - Star Node
- Topology
    - Curve of Point Node
    - Offset Point in Curve Node
    - Points of Curve Node

Read Curve Nodes

- Curve Handle Positions Node
- Curve Length Node
- Curve Tangent Node
- Curve Tilt Node
- Endpoint Selection Node
- Handle Type Selection Node
- Is Spline Cyclic Node
- Spline Length Node
- Spline Parameter Node
- Spline Resolution Node

Curve Handle Positions Node



Gets the two handle positions of each control point in a Bézier spline.

You can use the Set Handle Positions Node to change these positions.

Inputs

Relative
: Output the handle positions relative to the control point instead of in the local space of the geometry.

Properties

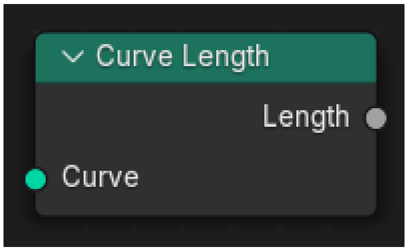
This node has no properties.

Outputs

Left
: The position of the control point's left handle.

Right
: The position of the control point's right handle.

Curve Length Node


Curve Length
Length
Curve

The *Curve Length* node outputs the length of all splines added together.

Inputs

Curve
: Standard geometry input.

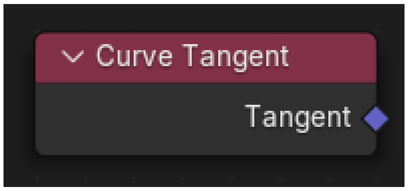
Properties

This node has no properties.

Output

Length
: Accumulated length of all splines of the curve.

Curve Tangent Node



The *Curve Tangent* node outputs the direction that a curve points in at each control point, depending on the direction of the curve (which can be controlled with the Reverse Curve Node). The output values are normalized vectors.

Warning

For NURBS and Bézier spline curves, keep in mind that the value retrieved from this node is the value at every control point, which may not correspond to the visible *evaluated* points. For example, a Bézier spline might have 48 evaluated points, but only four control points, if its resolution is 12. For NURBS splines the difference may be even more pronounced and the result may not be as expected. A Resample Curve Node can be used to create a poly spline, where there is a control point for every evaluated point.

Inputs

This node has no inputs.

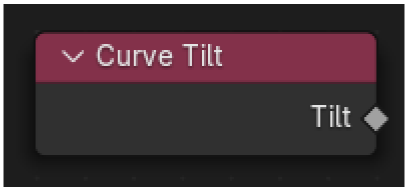
Properties

This node has no properties.

Outputs

Tangent
    The direction of the curve at every control point.

Curve Tilt Node



The *Curve Tilt* node outputs the angle used to turn the curve normal around the direction of the curve tangent in its evaluated points. Keep in mind that the output is per control point, just like the values that can be controlled in curve Edit Mode. For NURBS and Bézier splines, the values will be interpolated to the final evaluated points.

The input node for this data is the Set Curve Tilt node.

Inputs

This node has no inputs.

Properties

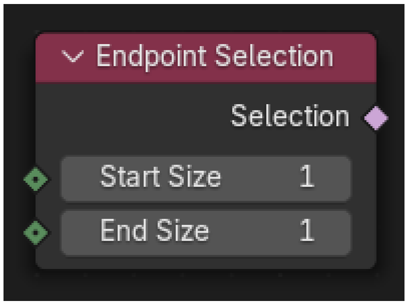
This node has no properties.

Outputs

Tilt

The tilt angle for the normal in radians.

Endpoint Selection Node



The *Endpoint Selection* node provides a selection for an arbitrary number of endpoints in each spline in a curve.

Note

The selection operates for every control point. This may not correspond to the evaluated points displayed in the viewport for NURBS and Bézier splines, where one control point may correspond to many evaluated points.

Tip

To use this data after the curve has been converted to another data type like mesh or a point cloud, the Capture Attribute Node can be used.

Inputs

Start Size
: The number of points to select from the start.

End Size
: The number of points to select from the end.

Properties

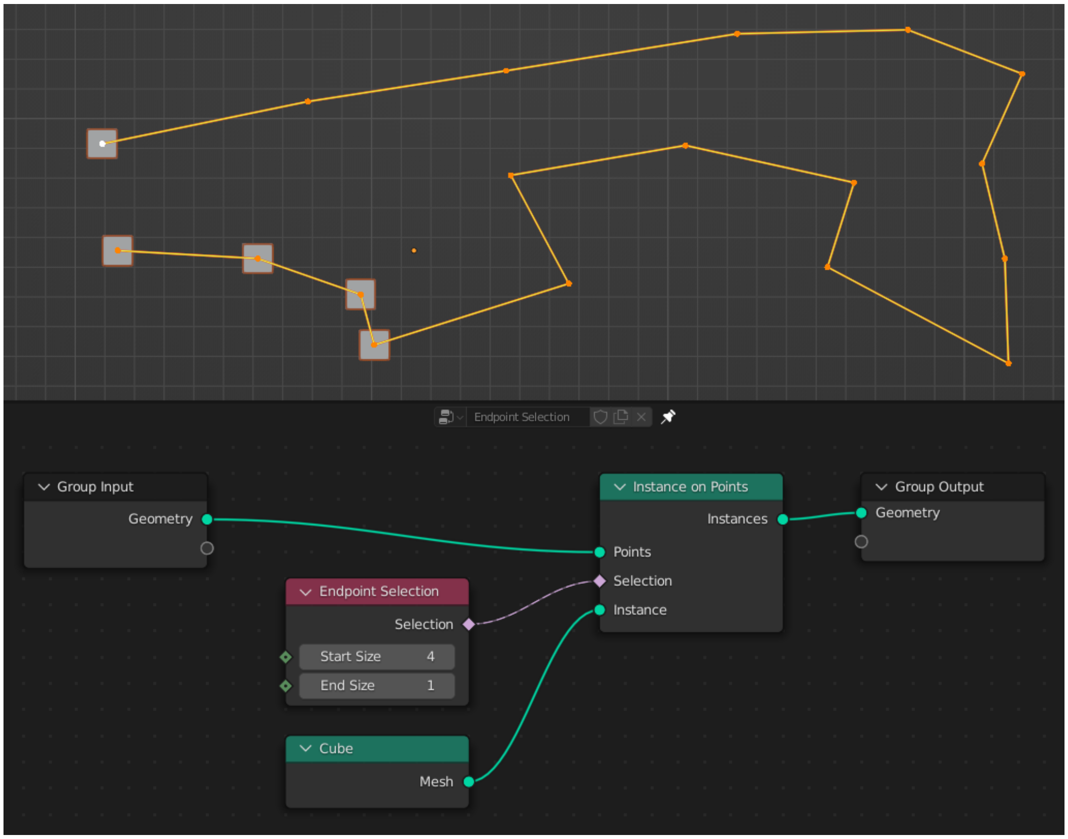
This node has no properties.

Outputs

Selection

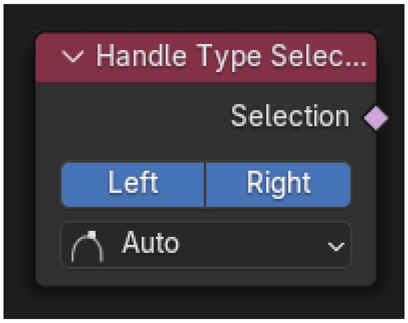
Selection of the start and end of each spline of the curve.

Examples


Endpoint Selection
Group Input
Geometry
Instance on Points
Instances
Points
Selection
Instance
Group Output
Geometry
Endpoint Selection
Selection
Start Size 4
End Size 1
Cube
Mesh

Anywhere the geometry is a curve, this node can be used to generate a selection of only the first and last points of each spline. Here, the *Points* input of the Instance on Points Node is a curve consisting of the poly spline shown in Edit Mode.

Handle Type Selection Node



Creates a selection based on the handle types of the control points.

The handle type of each control point can be changed with the Set Handle Type Node.

Inputs

This node has no inputs.

Properties

Mode

Whether to consider left and/or right handles. When both are selected, the output value is true if either of the handles are of the chosen type.

Left::	Consider the left handle.
Right::	Consider the right handle.

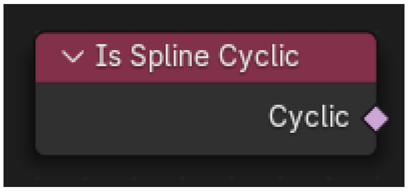
Handle Type

Handle type for which the selection will be true. See the Bézier curves page for more details on the different handle types.

Outputs

Selection
  Boolean field set to true wherever the handle type matches.

Is Spline Cyclic Node



The *Is Spline Cyclic* controls whether each of the curve splines start and endpoints form a connection. Its output corresponds to the built-in `cyclic` attribute on the curve spline domain.

The node to set this data is the Set Spline Cyclic Node.

Inputs

This node has no inputs.

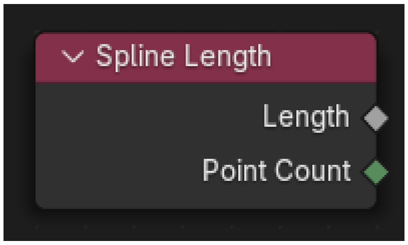
Properties

This node has no properties.

Outputs

Cyclic
    Whether the spline is cyclic.

Spline Length Node



The *Spline Length* node outputs the total length of each spline, as a distance, or a number of points. This is different than the Curve Length node, which adds up the total length for all of the curve's splines.

The output values correspond to the spline domain, but the node can be used to output a value for every curve control point as well.

Inputs

This node has no inputs.

Properties

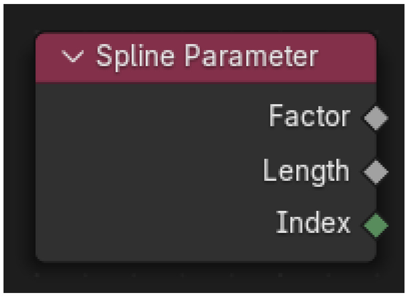
This node has no properties.

Outputs

Length
  The length of each spline.

Point Count
  The number of control points in each spline.

Spline Parameter Node



The *Spline Parameter* node outputs how far along each spline a control point is. The *Factor* output is different from dividing the index by the total number of control points, because the control points might not be equally spaced along the curve.

The first value is zero, so the output corresponds to the length at the control point rather than including the length of the following segment.

When used on the spline domain, the node outputs the portion of the total length of the curve (including all splines) has been traversed at the start of each spline. The order of the curve's splines is visible in the Spreadsheet Editor.

Warning

For NURBS and Bézier spline curves, keep in mind that the value retrieved from this node is the value at every control point, which may not correspond to the visible *evaluated* points. For NURBS splines the difference may be even more pronounced and the result may not be as expected. A Resample Curve Node node can be used to create a poly spline, where there is a control point for every evaluated point.

Note

When the *Length* is zero, the *Factor* is arbitrary. In this case the result is exceptionally calculated dividing the index by the total number of control points or curves.

Inputs

This node has no inputs.

Properties

This node has no properties.

Outputs

Factor

When the node is used on the point domain, the value is the portion of the spline's total length at each control point. On the spline domain it is the portion of the curve's total length at the start of the spline.

Length

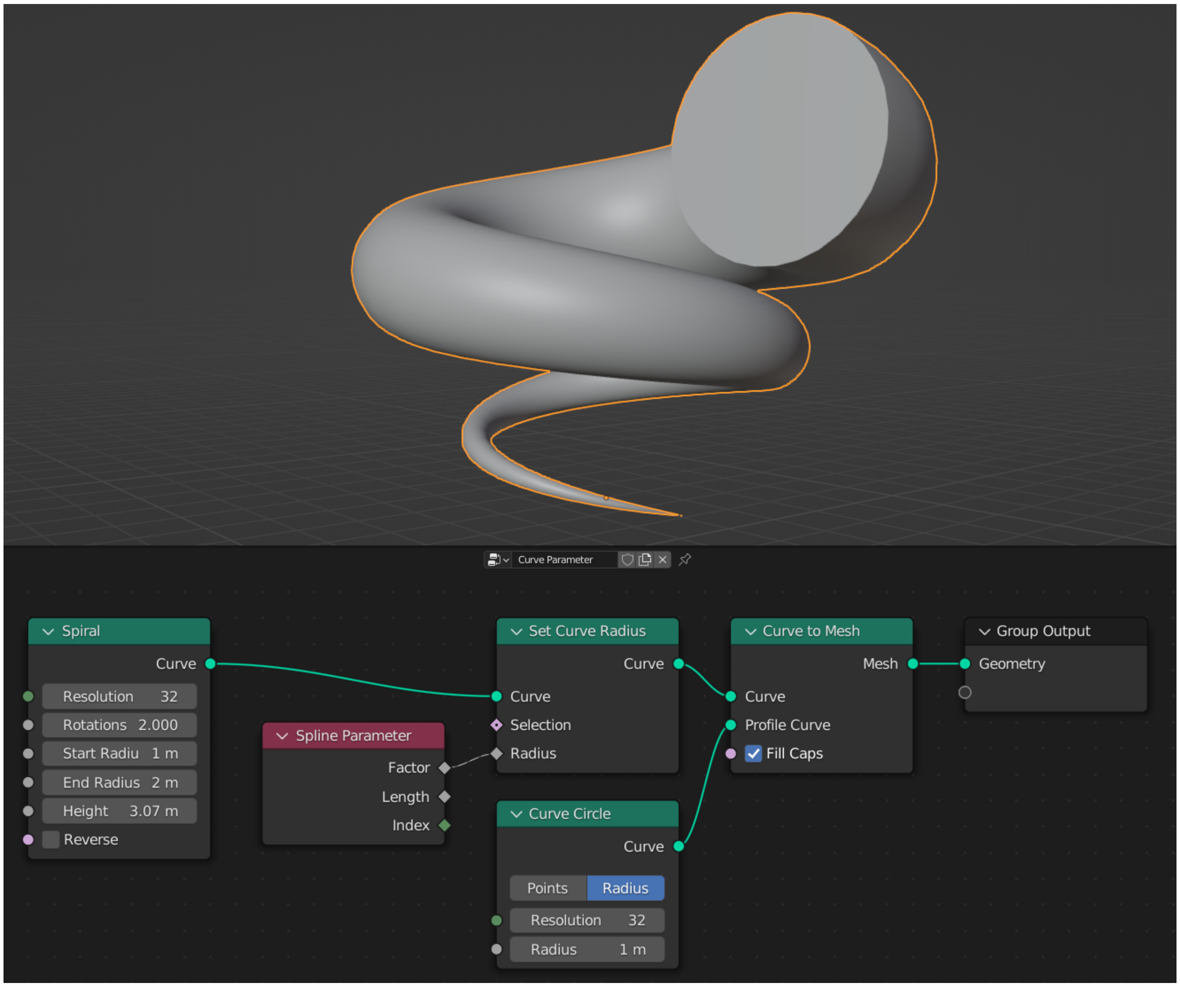
When the node is used on the point domain, the value is the distance along the spline to each control point. On the spline domain it is the length along the entire curve at the start of the spline.

Index

Each control point's index on its spline. This is different from the output of the index node, which also counts the control points in all previous splines.

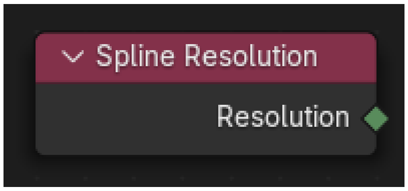
Examples

The parameter used to control the radius of the curve. The beginning of the spline has a radius of 0, the end has a radius of 1.


Curve Parameter
Spiral
Curve
Reverse
Spline Parameter
Factor
Length
Index
Set Curve Radius
Curve
Curve
Selection
Radius
Curve Circle
Curve
Points
Radius
Curve to Mesh
Mesh
Curve
Profile Curve
Group Output
Geometry

Resolution 32
Rotations 2.000
Start Radiu 1 m
End Radius 2 m
Height 3.07 m
☐ Reverse
Resolution 32
Radius 1 m
☑ Fill Caps

Spline Resolution Node



The *Spline Resolution* outputs the number of evaluated curve points that will be generated for every control point on the spline. This node works for NURBS, Bézier, and Catmull Rom splines.

For poly splines, there is a one-to-one correspondence between original points and evaluated points, so the resolution does not have an effect.

On Bézier splines, the resolution does not have an effect on segments between vector handles, since there are no extra evaluated points between the neighboring control points.

The node to set this data is the Set Spline Resolution Node.

Inputs

This node has no inputs.

Properties

This node has no properties.

Outputs

Resolution

The integer resolution value for each spline.

Curve Sample Nodes

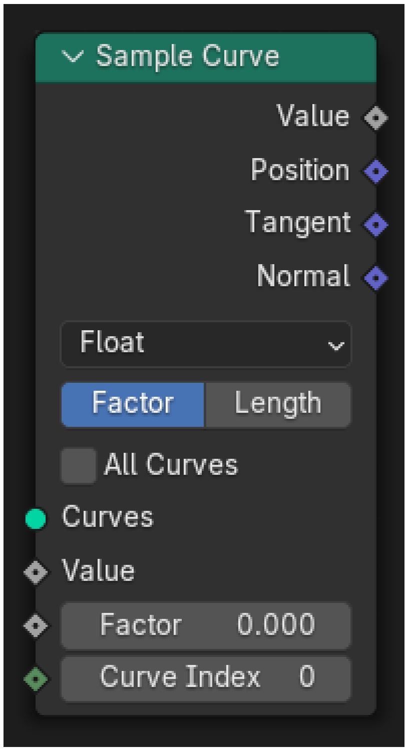
- Sample Curve Node

Sample Curve Node

The *Sample Curve* calculates a point on a curve at a certain distance from the start of the curve, specified by the length or factor inputs. It also outputs data retrieved from that position on the curve. The sampled values are linearly interpolated from the values at the evaluated curve points at each side of the sampled point.

Note

When the curve contains multiple splines, the sample position is found based on the total accumulated length, including the lengths of all previous splines. The order of the splines is the same order as displayed in the Spreadsheet Editor.


Sample Curve
Value
Position
Tangent
Normal
Float
Factor
Length
All Curves
Curves
Value
Factor 0.000
Curve Index 0

Inputs

Curves
: Standard geometry input with a curve component.

Value
: A field input to evaluate custom attributes. The evaluation is outputted through the *Value* output.

Factor *Factor mode*
: The portion of the total length used to determine the sample position.

Length *Length mode*

A length in distance units used to determine how far along the curve to travel before sampling.

Curve Index

An index to only evaluate specific splines, these indices can be specified manually or from the Index Node. This input is ignored when the *All Curves* property is enabled.

Properties

Data Type

The data type used for the evaluated data.

Mode

How to find endpoint positions for the trimmed spline. The option acts the same as it does in the Trim Curve Node.

Factor::	Find the endpoint positions using a factor of each spline's length. The input values should be between 0 or 1.
Length::	Find the endpoint positions using a length from the start of each spline. The input values should be between 0 and the length of the splines.

All Curves

Sample lengths based on the total length of all curves, rather than using a length inside each selected curve.

Outputs

Value

The value of the input *Value* at the sample point.

Position

The position at the sample along the spline.

Tangent

The normalized curve tangent at the sample.

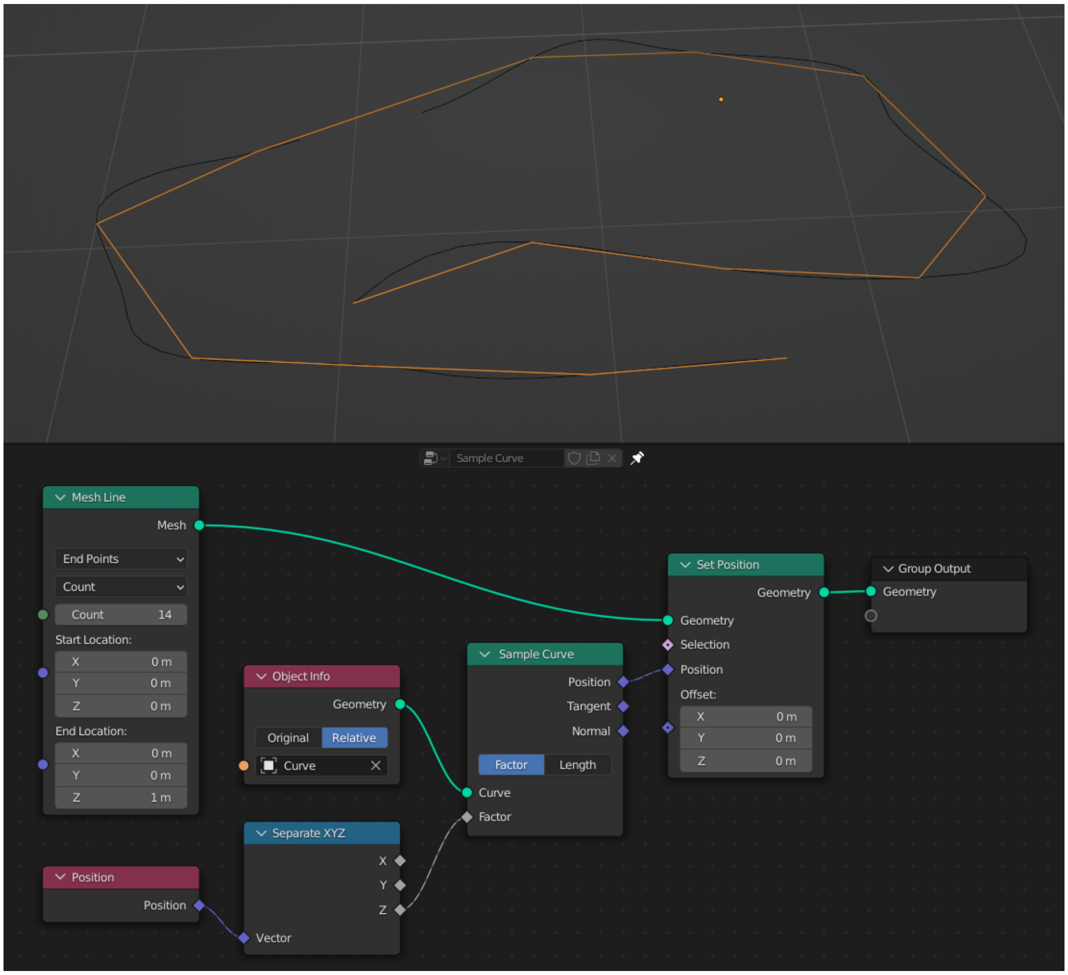
Tip

This output can be combined with the Align Rotation to Vector Node to create a rotation that lines up with direction of the curve. Including the *Normal* output in a second align node after can align the other rotation axis.

Normal

The normalized curve normal at the sample.

Examples

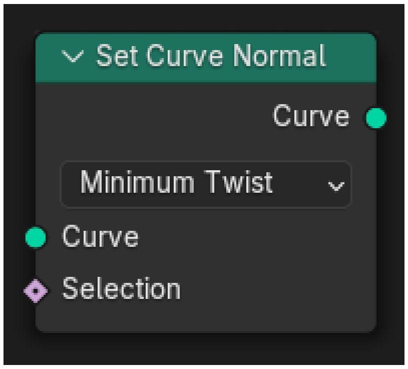

Sample Curve
Mesh Line
Mesh
End Points
Count
Count 14
Start Location:
X 0 m
Y 0 m
Z 0 m
End Location:
X 0 m
Y 0 m
Z 1 m
Object Info
Geometry
Original
Relative
Curve
Sample Curve
Position
Tangent
Normal
Factor
Length
Curve
Factor
Set Position
Geometry
Geometry
Selection
Position
Offset:
X 0 m
Y 0 m
Z 0 m
Group Output
Geometry
Separate XYZ
X
Y
Z
Vector
Position
Position

Here, the *Count* mode of the Resample Curve Node is recreated, except a mesh is used for the result instead of a curve. The Z axis of the position can be used as the sample factor because the position is between zero and one for the entire line.

Write Curve Nodes

- Set Curve Normal Node
- Set Curve Radius Node
- Set Curve Tilt Node
- Set Handle Positions Node
- Set Handle Type Node
- Set Spline Cyclic Node
- Set Spline Resolution Node
- Set Spline Type Node

Set Curve Normal Node



The *Set Curve Normal* controls the method used to calculate curve normals for every curve.

The node doesn't set the normals directly, those are calculated later as necessary. Combined with the tilt attribute value at each control point, this will define the final normals accessible with the Normal Node.

Internally this node adjusts the values of the `normal_mode` attribute on each curve.

Inputs

Curve

Standard geometry input, containing curves.

Selection

Whether or not to change the value on each curve.

Normal

Input for the custom normal attribute (`custom_normal`) when using *Free* mode.

Properties

Mode

The method for evaluation of the curve's normals

Minimum Twist:: The final normals are calculated to have the smallest twist around the curve tangent across the whole curve.

Z-Up::	The final normals are calculated so that they are perpendicular to the Z axis and the tangent. If a series of points is vertical, the X axis is used.
Free::	Use the stored custom normal attribute (`custom_normal`) as the final normals. This mode adds a *Normal* input that can be used to set the value of the custom normal.

Note

Custom normals are not rotation invariant, meaning normals must be set **after** any rotation transformations; i.e. at the end of the node tree or at the bottom of the modifier stack.

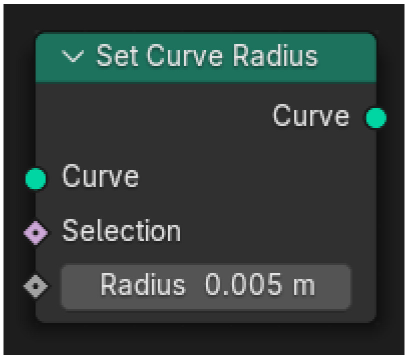
Outputs

Curve

Standard geometry output.

Set Curve Radius Node

The *Set Curve Radius* controls the radius of the curve, used for operations like the size of the profile in the Curve to Mesh node. The value is set for every control point, and is then interpolated to each evaluated point in between the control points.



The input node for this data is the Radius node.

Inputs

Curve
: Standard geometry input, containing curves.

Selection
: Whether or not to change the value on each control point. True values mean the value will be changed, false values mean the value will remain the same.

Radius
: The radius value for each control point.

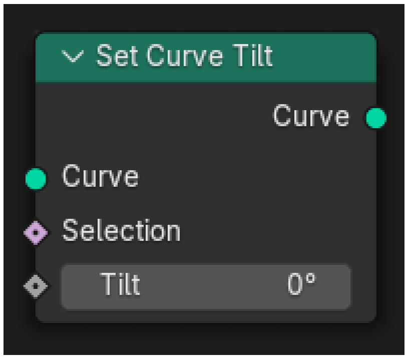
Properties

This node has no properties.

Outputs

Geometry
: Standard geometry output.

Set Curve Tilt Node



The *Set Curve Tilt* controls the tilt angle at each curve control point. That angle rotates normal vector which is generated at each point when evaluating the curve. The normal then can be retrieved with the Normal Node.

The rotation of the normal vector is an Axis Angle rotation. It is the same as the Vector Rotate Node operation with the tangent vector as the axis, the raw evaluated normal is used as the original vector, and the tilt as the rotation angle.

The input node for this data is the Curve Tilt Node.

Inputs

Curve
: Standard geometry input, containing curves.

Selection
: Whether or not to change the value on each control point. True values mean the value will be changed, false values mean the value will remain the same.

Tilt
: The tilt angle for each control point.

Properties

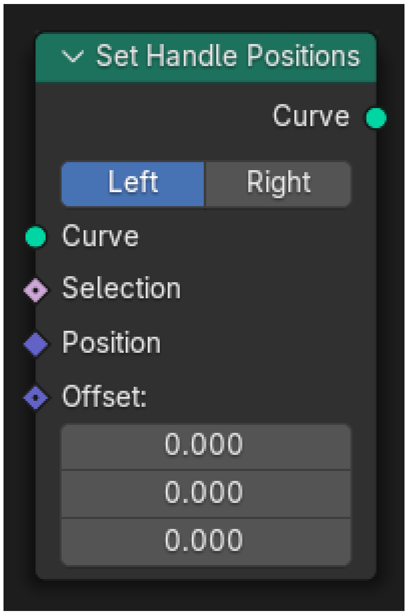
This node has no properties.

Outputs

Curve

Standard geometry output.

Set Handle Positions Node



The *Set Handle Positions* node sets the positions for the handles of Bézier curves. They can be used to alter the generated shape of the curve. The input node for this data is the Curve Handle Positions Node. See the Bézier curves page for more details.

Note

When the position is changed, *Auto* handle types will be converted to *Aligned*, and *Vector* handle types will be converted to *Free*.

Note

The left and right handles cannot be changed at the same time with this node. That is because it would break the alignment for left and right handles at the same control point.

Inputs

Curve
: Standard geometry input, containing curves.

Selection
: Whether or not to change the handle position on each control point. True values mean the resolution will be changed, false values mean the resolution will remain the same.

Position

The new handle position.

Note

The handle positions are the global position of the handle, they are not relative to the position of the corresponding control point.

Offset

An optional translation for each handle. This is evaluated at the same time as the *Position* input, meaning that fields evaluated for it will not reflect the changed position.

Properties

Left / Right

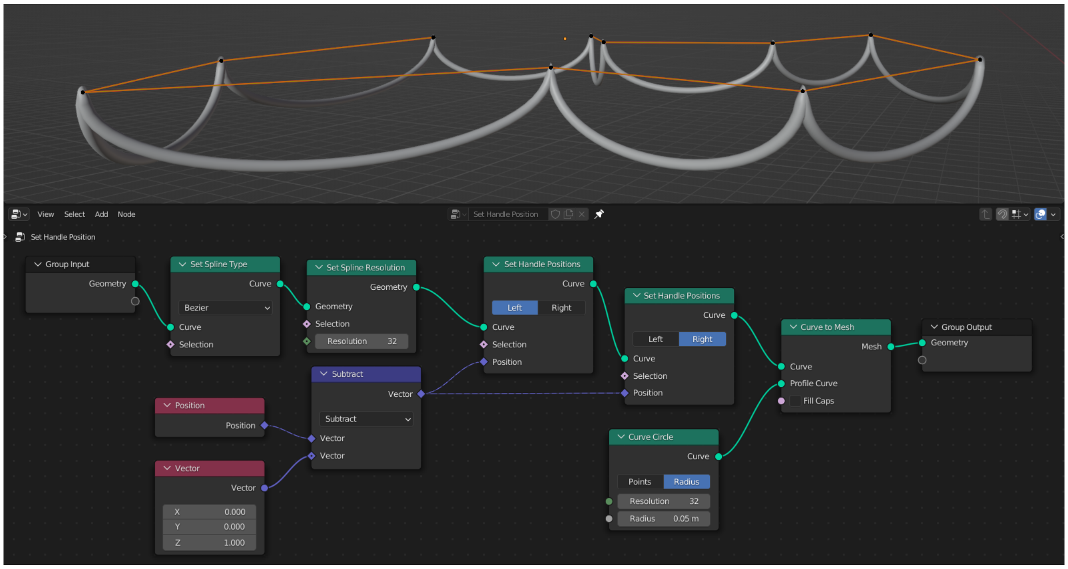
Whether to set the handle position of the left or right handle. The *Left* handle is closer to the start of the spline, and the *Right* handle is closer to the end.

Outputs

Curve

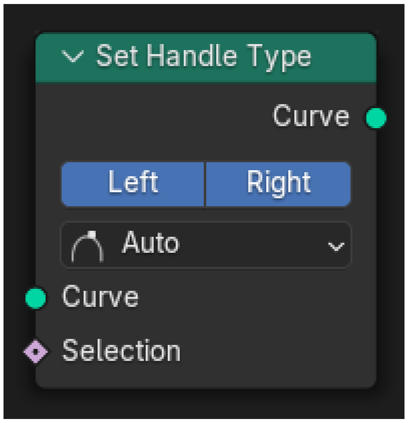
Standard geometry output.

Examples


View
Select
Add
Node
Set Handle Position
Set Handle Position
Group Input
Geometry
Set Spline Type
Curve
Bezier
Curve
Selection
Set Spline Resolution
Geometry
Geometry
Selection
Resolution 32
Set Handle Positions
Curve
Left
Right
Curve
Selection
Position
Set Handle Positions
Curve
Left
Right
Curve
Selection
Position
Curve to Mesh
Mesh
Curve
Profile Curve
Fill Caps
Group Output
Geometry
Position
Position
Subtract
Vector
Subtract
Vector
Vector
Vector
Vector
X 0.000
Y 0.000
Z 1.000
Curve Circle
Curve
Points
Radius
Resolution 32
Radius 0.05 m

Here, the handles are adjusted to the same position as the control points, but offset down in the Z direction slightly. With the Set Spline Type Node, the curve can be a poly spline in Edit Mode, to make adjustment simpler for the user of the node group.

Set Handle Type Node



Sets the handle type for the points on the Bézier curve that are in the selection.

A selection for a certain handle type can be retrieved with the Handle Type Selection Node.

Inputs

Curve
Bézier curve.

Selection
The points whose handle types will be changed.

Properties

Mode
Whether to update left or right or both handles. When both are selected, both the left and the right handles will be updated.

Left::	Update the left handles.
Right::	Update the right handles.

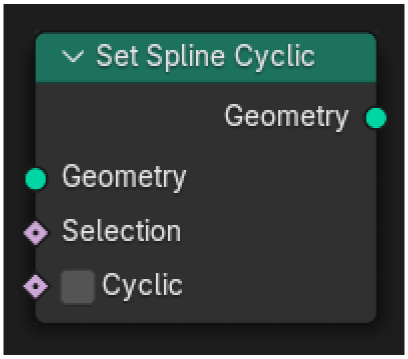
Handle Type
The handle type to switch to. See the Bézier curves page for more details on the different handle types.

Outputs

Curve

Standard curve output.

Set Spline Cyclic Node



The *Set Spline Cyclic* controls whether each spline will loop back on itself. Each spline has the same number of control points whether or not it is set as cyclic. But when displaying in the viewport or for operations with other nodes, a connection will be made between the first and last control points.

The input node for this data is the Is Spline Cyclic Node.

Inputs

Curve
Standard geometry input.

Selection
Whether or not to change the cyclic value on each spline. True values mean the value will be changed, false values mean the value will remain the same.

Cyclic
Whether or not to join the first and last points of each spline.

Properties

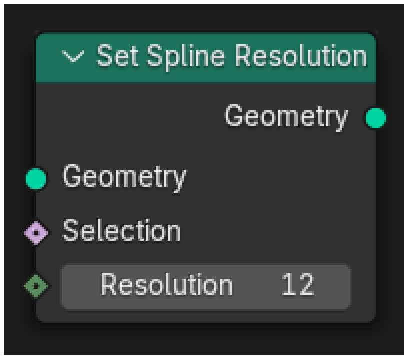
This node has no properties.

Outputs

Curve

Standard geometry output.

Set Spline Resolution Node



The *Set Spline Resolution* node sets the value for how many evaluated points should be generated on the curve for every control point. It only has an effect on NURBS, Bézier, and Catmull Rom splines. In case of Bézier splines, the resolution does not have an effect on segments between vector handles.

The evaluated points are displayed in the viewport, used in the Curve to Mesh Node node, and optionally used in the Resample Curve Node.

The input node for this data is the Spline Resolution Node.

Inputs

Curve
    Standard geometry input.

Selection
    Whether or not to change the resolution value on each spline. True values mean the value will be changed, false values mean the resolution will remain the same.

Resolution
    The number of evaluated points generated for each control point in NURBS and Bézier splines. It controls the accuracy of operations like trimming or sampling a curve. Higher resolutions are more accurate, but slower.

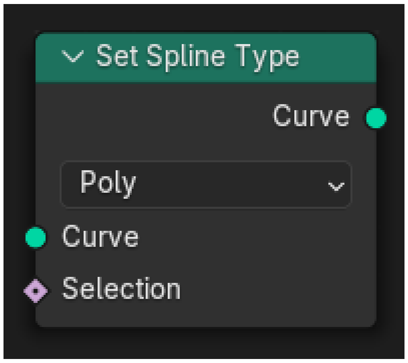
Properties

This node has no properties.

Outputs

Curve
: Standard geometry output.

Set Spline Type Node



Sets the spline type for the splines in the curve component that are in the selection.

Inputs

Curve
: Standard geometry input with a curve component.

Selection
: The splines whose spline types will be changed.

Properties

Spline Type
: The type to convert the splines in the selection to. Read the [Spline Types](#) page for more details on the different spline types.

Bézier::	Convert to a Bézier spline. A spline converted from a poly spline gets vector handles, while one converted from NURBS or Catmull Rom spline gets auto handles. **Note** When converting from a NURBS spline to a Bézier spline, at least six points are needed. When the number of points is not a multiple of

	three a full conversion is not possible and the spline has to be truncated.
NURBS::	Convert to a NURBS spline.
Poly::	Convert to a poly spline.
Catmull Rom::	Convert to a Catmull Rom spline.

Outputs

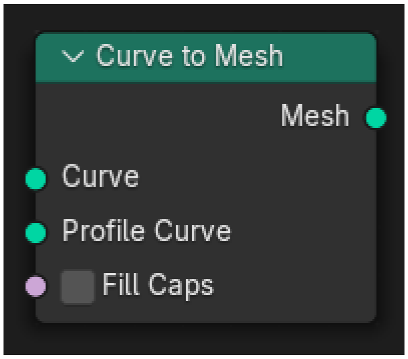
Curve

Standard curve output.

Curve Operation Nodes

- Curve to Mesh Node
- Curve to Points Node
- Curves to Grease Pencil Node
- Deform Curves on Surface Node
- Fill Curve Node
- Fillet Curve Node
- Grease Pencil to Curves Node
- Interpolate Curves Node
- Merge Layers Node
- Resample Curve Node
- Reverse Curve Node
- Subdivide Curve Node
- Trim Curve Node

Curve to Mesh Node



The Curve to Mesh node converts all splines of a curve to a mesh. Optionally, a profile curve can be provided to give the curve a custom shape. This profile curve is scaled by the radius at each control point; see the Set Curve Radius Node.

The node transfers attributes to the result. Attributes that are built-in on meshes but not curves, like `sharp_face`, will be transferred to the correct domain as well.

> **Tip**
>
> The output mesh has sharp edges set from the profile curve tagged automatically. If any splines in the profile curve are Bézier splines and any of the control points use *Free* or *Vector* handles, the corresponding edges will be shaded sharp.

Inputs

Curve
: Standard geometry input. All non-curve components are ignored.

Profile Curve

If a profile curve is provided, it will be extruded along all splines. Otherwise the generated mesh will just be a chain of edges.

Fill Caps

If the profile spline is cyclic, fill the ends of the generated mesh with n-gons. The resulting mesh is Manifold, the two new faces for each spline are simply connected to existing edges.

Properties

This node has no properties.

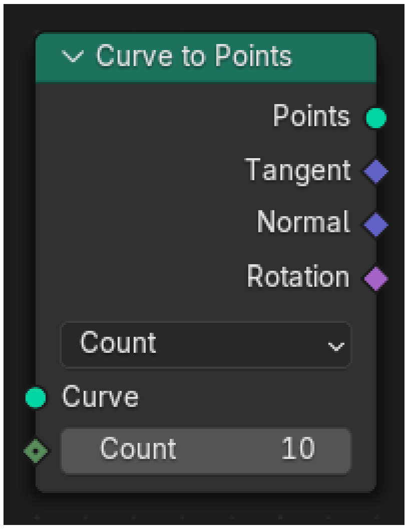
Outputs

Mesh

Standard geometry output.

Curve to Points Node

The *Curve to Points* node generates a point cloud from a curve.


Curve to Points
Points
Tangent
Normal
Rotation
Count
Curve
Count 10

Inputs

Curve
: Standard curve input.

Count
: Number of points generated. This input is only available for *Count* mode.

Length
: Length of the curve. This input is only available for *Length* mode.

Properties

Mode

Evaluated::	Creates points from the curve’s evaluated points based on the resolution attribute for NURBS and Bézier splines. This mode will generally be the fastest, since the second step of sampling equal lengths is not necessary.
Count::	Sample each spline by evenly distributing the specified number of points.
Length::	Sample each spline by splitting it into segments with specified length. The length will be rounded down so that a whole number of samples will fit in each input spline. To counteract jumping when the length of the spline changes, the Trim Curve Node can be used with a multiple of this length.

Outputs

Points
Generated point cloud.

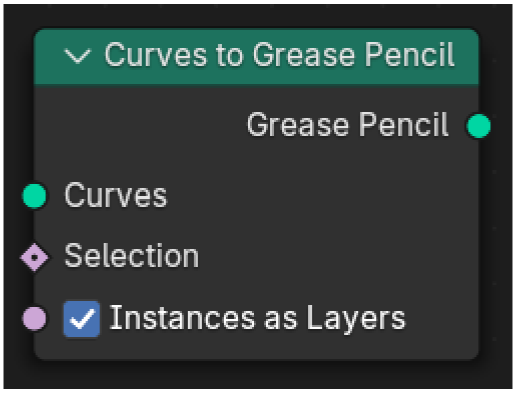
Tangent
The normalized curve tangent at the sampled position, or the direct evaluated normal in *Evaluated* mode.

Normal
The normal value from the evaluated curve at each result point. This is the same value from the Normal Node at those positions.

Rotation
The Euler rotation build from the *Tangent* and *Normal* outputs, for convenience.

Curves to Grease Pencil Node

The *Curves to Grease Pencil* node Converts top-level curve instances into Grease Pencil layers


Curves to Grease Pencil
Grease Pencil
Curves
Selection
☑ Instances as Layers

Inputs

Curves
Either plain curves or curve instances.

Selection
Either a curve or instance selection.

Instances as Layers
Create a separate layer for each instance. If instances are used, realized curve geometry will be ignored. Layer names will use the instance geometry name. If real curve geometry is used, a single layer is created with the input geometry's name.

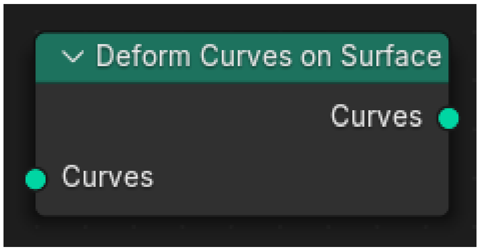
Properties

This node has no properties.

Outputs

Grease Pencil
Standard Grease Pencil geometry.

Deform Curves on Surface Node



The *Deform Curves on Surface* node translates and rotates each curve based on the difference in its root position. The root position is defined by UV coordinates stored on each curve and the UV Map selected for the purpose in the Curves surface settings.

The transformation is calculated based on the difference of the original mesh (before shape keys and modifiers are evaluated), and the final mesh.

Unlike other geometry nodes, this node has quite a few implicit inputs:

- The original and evaluated mesh are retrieved from the modifier object's surface property. This means the node only works for curves objects.
- The original and evaluated UV map are also retrieved from the object's surface property.
- A 3D vector attribute named `rest_position`, used for calculating tangents for rotating curves that are consistent with the tangents calculated on the original mesh (the rotation needs to be calculated from the normal and tangent of the original and evaluated meshes).
- A 2D vector attribute on the curve domain named `surface_uv_coordinate` to store the location of the root positions on the surface mesh's UV map.

In future development, this node will be generalized so the setup is more flexible.

Parts of the internal operation are similar to the Sample UV Surface Node.

Warning

In order to achieve consistent deformation after the Subdivision Surface Modifier, the *UV Smooth* option of the modifier should be set to *None*. Otherwise the surface UV map will be subdivided in a way that may invalidate the curve UV attachement points stored in the `surface_uv_coordinate` attribute.

Inputs

Curves
Standard curves input.

Properties

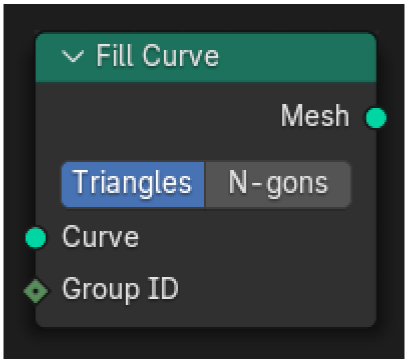
This node has no properties.

Outputs

Curves
Standard curves output.

Fill Curve Node

The *Fill Curve* node generates a mesh using the constrained Delaunay triangulation algorithm with the curves as boundaries. The mesh is only generated flat with a local Z of 0.



Inputs

Curve
: Standard geometry input with a curve component.

Group ID
: Value used to group curves together. Curves with different Group ID are treated separately.

Properties

Mode
: The type of geometry the output consists of.

Triangles::	The output is made up of triangles.
N-gons::	The output is made up of n-gons.

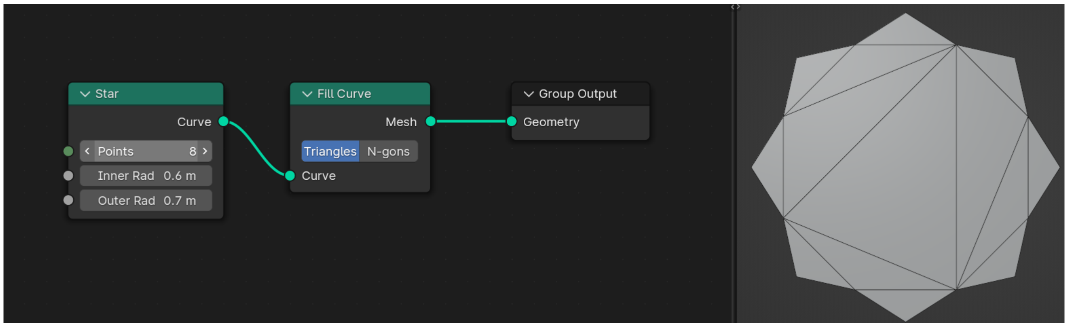
Outputs

Mesh
: The filled-in curves.

Examples

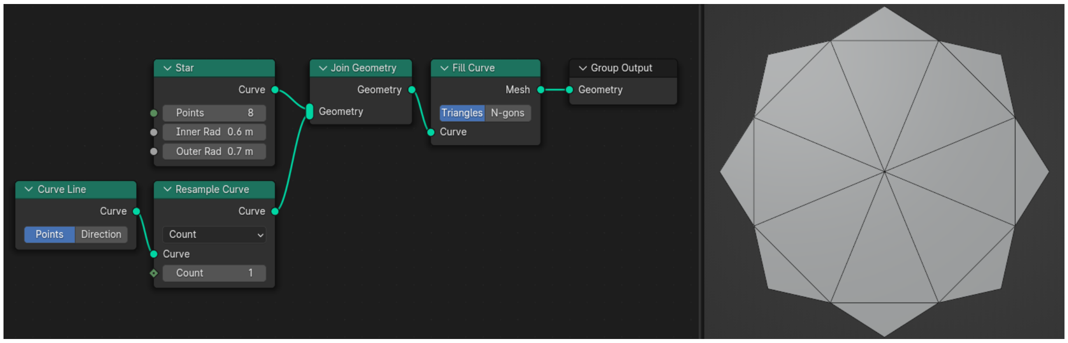
Customized triangulation

One or many “single point spline” can be used to customize the triangulation of the filled-in curves.


Star
Curve
Fill Curve
Mesh
Curve
Group Output
Geometry

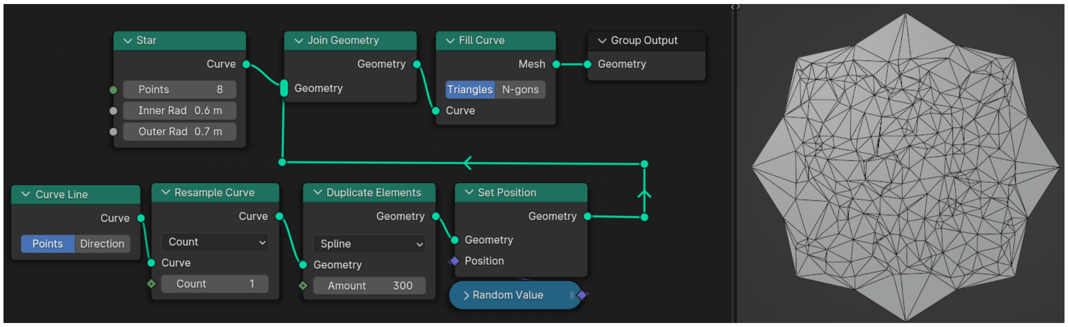
Points 8
Inner Rad 0.6 m
Outer Rad 0.7 m
Triangles N-gons

This is the default behavior of the *Fill Curve* node applied to the star primitive.


Star
Curve
Join Geometry
Geometry
Geometry
Fill Curve
Mesh
Curve
Group Output
Geometry
Curve Line
Curve
Resample Curve
Curve
Curve

Points 8
Inner Rad 0.6 m
Outer Rad 0.7 m
Triangles N-gons
Points Direction
Count
Count 1

Here, a single curve point is joined to the star primitive to customize the triangulation.

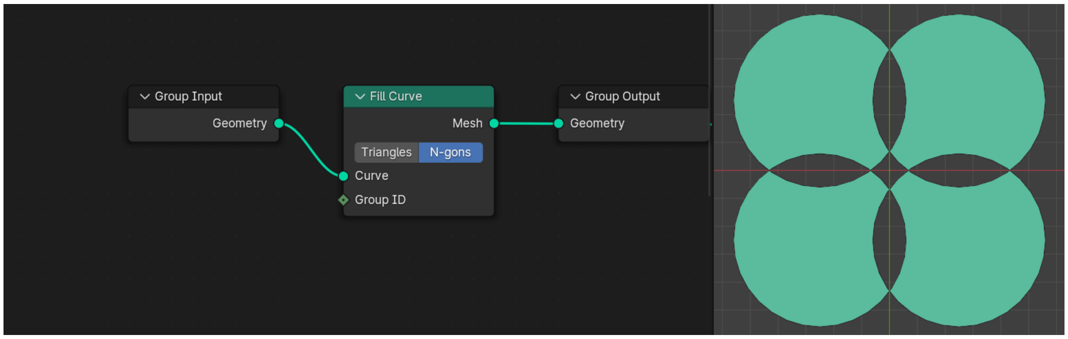

Star
Curve
Join Geometry
Geometry
Geometry
Fill Curve
Mesh
Curve
Group Output
Geometry
Curve Line
Curve
Resample Curve
Curve
Curve
Duplicate Elements
Geometry
Geometry
Set Position
Geometry
Geometry
Position
Random Value

Points 8
Inner Rad 0.6 m
Outer Rad 0.7 m
Triangles N-gons
Points Direction
Count
Count 1
Spline
Amount 300

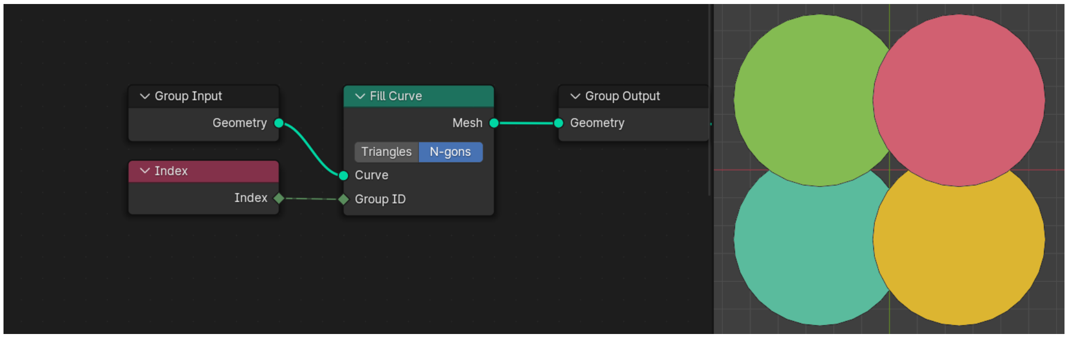
Here, 300 single curve point are joined to the star primitive to customize the triangulation.

Group ID

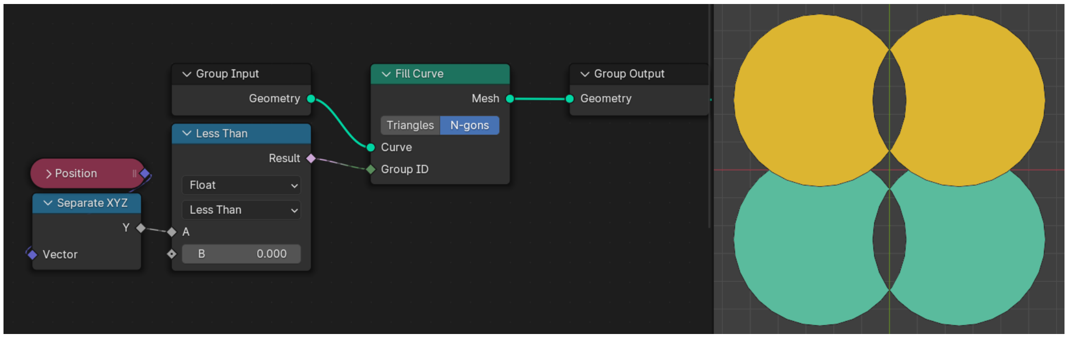
The following figures display diverse application of the Group ID.


Group Input
Geometry
Fill Curve
Mesh
Triangles
N-gons
Curve
Group ID
Group Output
Geometry

Here, the 4 curves share the same Group ID, resulting in 1 mesh island (default behavior).


Group Input
Geometry
Index
Index
Fill Curve
Mesh
Triangles
N-gons
Curve
Group ID
Group Output
Geometry

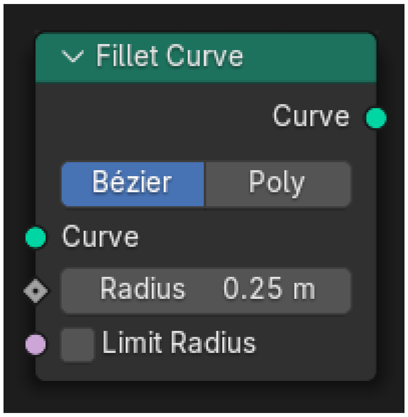
Here, the 4 curves have different Group ID, resulting in 4 mesh islands.


Group Input
Geometry
Less Than
Result
Float
Less Than
A
B 0.000
Position
Separate XYZ
Y
Vector
Fill Curve
Mesh
Triangles
N-gons
Curve
Group ID
Group Output
Geometry

Here, the 4 curves are seperated into two groups based on their positions, resulting in 2 mesh islands.

Fillet Curve Node

The *Fillet Curve* rounds corners on curve control points, similar to the effect of the Bevel Modifier on a 2D mesh. However, a key difference is that the rounded portions created by the Fillet Curve node are always portions of a circle.



Inputs

Curve

Standard geometry input with a curve component.

Radius

The radius of the circle portion generated at each fillet.

Count

Only in *Poly* mode, the number of control points to add for each fillet.

Limit Radius

Whether to limit the maximum value of the radius in order to avoid overlapping fillets.

Properties

Method

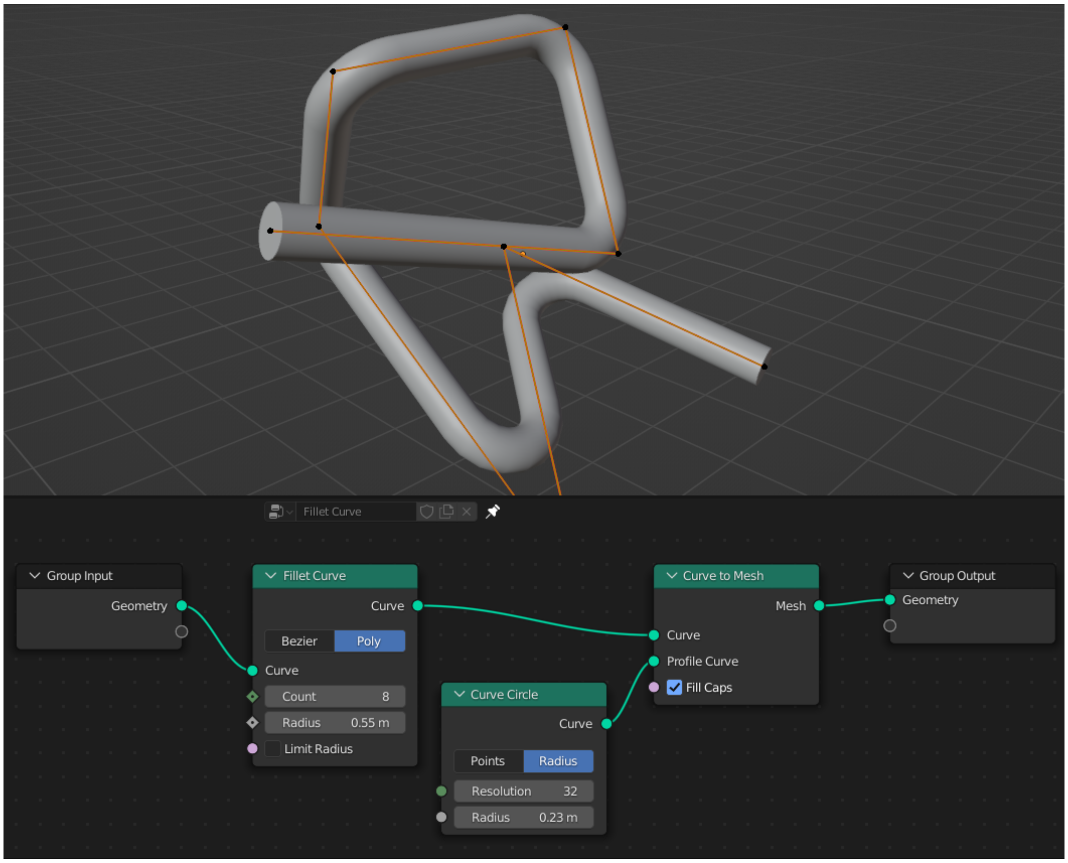
Bézier::	Only two control points will be generated for every filleted control point. The shape generated by the aligned handles on the generated control points on either side of the fillet is used to create the circle portion shape, meaning that the number of segments in the fillet shape depends on the [spline's resolution value]().
Poly::	The number of control points generated for each field input is controlled directly with an integer field input. This mode works better for poly and NURBS splines.

Outputs

Curve

Standard geometry input with a curve component.

Examples


Fillet Curve
Group Input
Geometry
Fillet Curve
Curve
Bezier
Poly
Curve
Curve to Mesh
Mesh
Curve
Profile Curve
Curve Circle
Curve
Points
Radius
Group Output
Geometry

Count 8

Radius 0.55 m

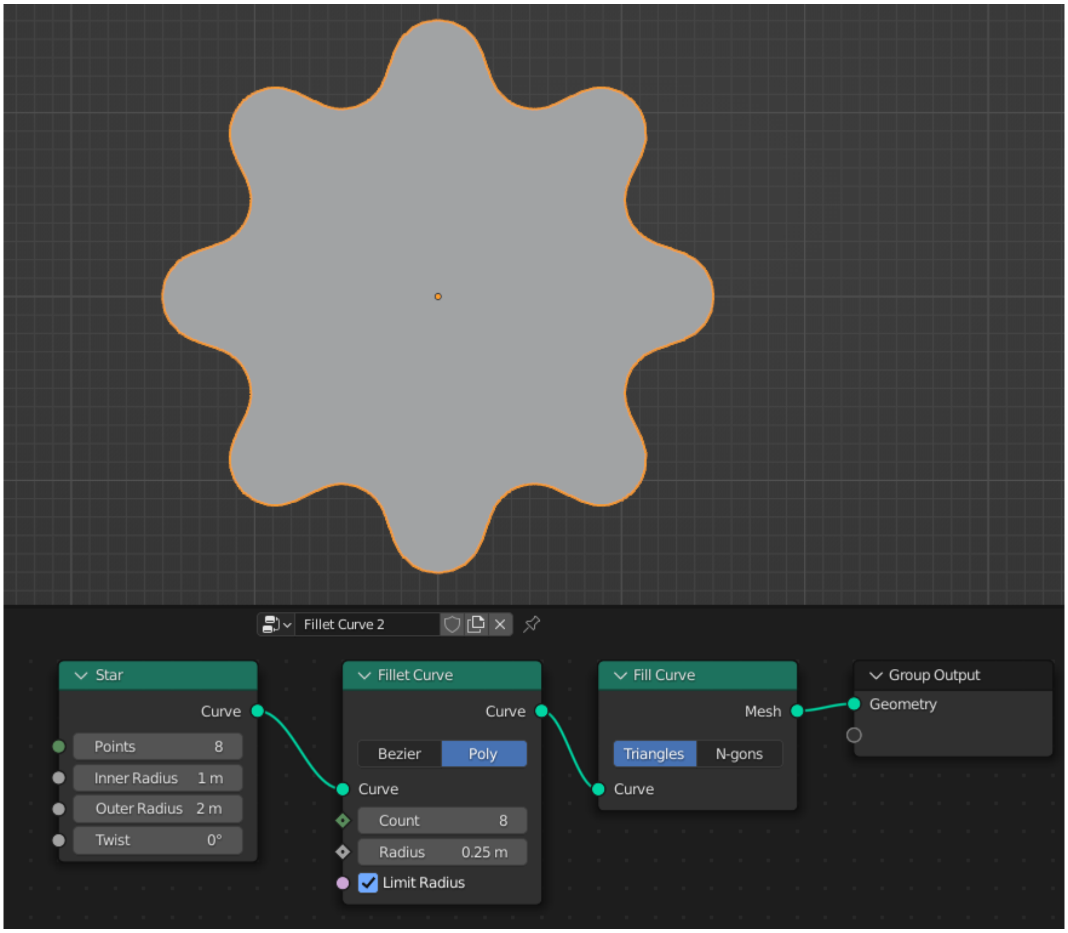
☐ Limit Radius

☑ Fill Caps

Resolution 32

Radius 0.23 m

The node can be used to round the corners of simple 3D poly splines.



Fillet Curve 2
Star
Curve
Fillet Curve
Curve
Bezier
Poly
Curve
Fill Curve
Mesh
Triangles
N-gons
Curve
Group Output
Geometry

Points 8

Inner Radius 1 m

Outer Radius 2 m

Twist 0°

Count 8

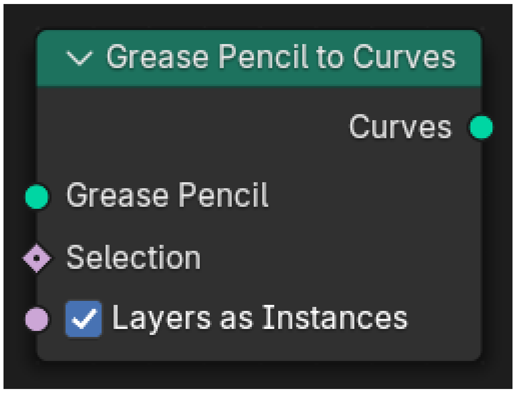
Radius 0.25 m

☑ Limit Radius

The node can be combined with the curve primitive nodes to make more interesting shapes.

Grease Pencil to Curves Node

The *Grease Pencil to Curves* node converts each Grease Pencil layer into an instance that contains curves.


Grease Pencil to Curves
Curves
Grease Pencil
Selection
☑ Layers as Instances

Inputs

Grease Pencil

Standard Grease Pencil geometry.

Selection

Selects the layers to convert.

Instances as Layers

Create a separate curve instance for every layer.

Tip

This is equivalent to a Realize Instances operation after creating the curve geometry instances. This can simplify some logic later on, but typically keeping this operation off will be much better for performance.

Properties

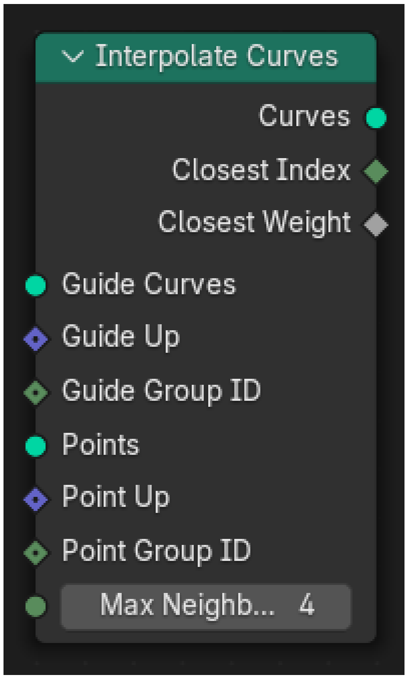
This node has no properties.

Outputs

Curves
    Standard curves geometry.

Interpolate Curves Node

Generate new curves on points by interpolating between existing curves. This is useful to have a smaller set of original curves to make editing easier and faster while still generating high-density curves for the viewport or a final render.


Interpolate Curves
Curves
Closest Index
Closest Weight
Guide Curves
Guide Up
Guide Group ID
Points
Point Up
Point Group ID
Max Neighb... 4

Inputs

Guide Curves

Base curves that new curves are interpolated between.

Guide Up

Optional up vector that is typically a surface normal. Supplying an up vector will improve the quality of the interpolation, making it aware of the surface shape which it otherwise wouldn't have a way to know about.

Tip

In a typical "child hair" generation setup, this up direction is retrieved with a combination of the Sample UV Surface Node using the same geometry that the points were distributed on, and the Normal Node.

Guide Group ID

Splits guides into separate groups. New curves interpolate existing curves from a single group. This input can be useful for generating hair parts by stopping curves in different sections (with different group IDs) from affecting each other.

Points

The positions of the first root control points of the newly generated interpolated curves.

Points Up

Optional up vector that is typically a surface normal.

Point Group ID

The curve group to interpolate in.

Max Neighbors

Maximum amount of close guide curves that are taken into account for interpolation.

Properties

This node has no properties.

Outputs

Curves

The guide curves with interpolated curves.

Closest Index

Index of the closest guide curve for each generated curve.

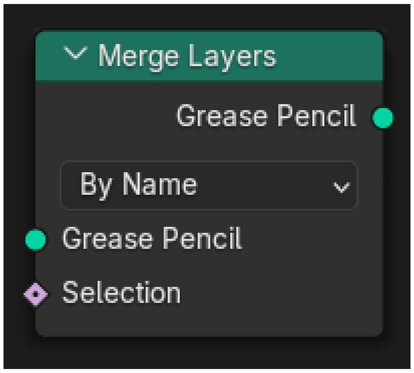
Note

Internally this node mixes the data from multiple guide curves, with the maximum number of sources depending on the *Max Neighbors* input. This output is only the index of the curve with the largest weight.

Closest Weight

Weight of the closest guide curve for each generated curve.

Merge Layers Node



Combines multiple [Grease Pencil Layers] into a single layer.

See also

[Merge Layers Operator]

Inputs

Grease Pencil
The grease pencil object with multiple layers to combine.

Selection
Operate of a subset of layers by setting the field layer index value to *true*. By default, all layers are selected.

Group ID
The index numbers of the layer to be merged. This input is only visible when using *By Group ID* Mode.

Properties

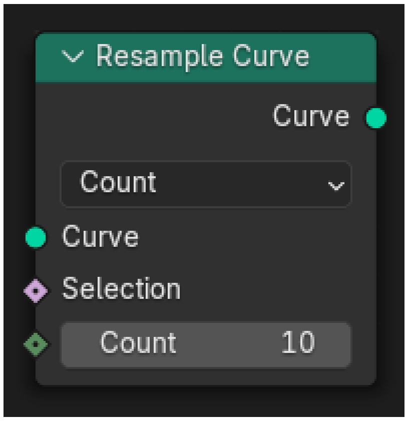
Mode
Determines how to choose which layers are merged.

By Name::	Combine all layers which have the same name.
By Group ID::	Provide a custom group ID for each layer and all layers with the same ID will be merged into one.

Outputs

Grease Pencil
: The grease pencil object with combined layers.

Resample Curve Node



The Resample Curve node creates a poly spline for each input spline. In the *Count* and *Length* modes, the control points of the new poly splines will have uniform spacing.

Tip

Use a field as an input to have a different count/length for each spline.

Inputs

Curve
: Standard geometry input.

Selection
: Whether or not to resample each spline. True values mean spline will be resampled to a poly spline, false values mean the spline will be unaffected.

Count
: The number of control points on the new splines.

Length
The approximate length between the control points of the new splines.

Tip

A Trim Curve Node can be used with a multiple of the input length to make the distance between each sampled point exact, even when the length of the spline changes.

Properties

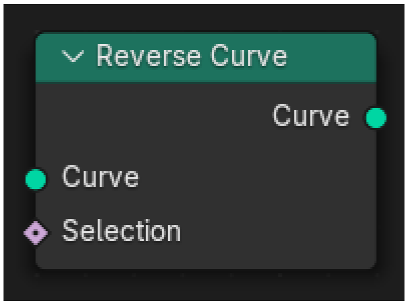
Mode
How to specify the amount of samples.

Count::	Sample the specified number of points along each spline.
Length::	Calculate the number of samples by splitting each spline into segments with the specified length. The length will be rounded down so that a whole number of samples will fit in each input spline.
Evaluated::	Evaluate the spline's points based on the resolution attribute for NURBS and Bézier splines. Changes nothing for poly splines.

Outputs

Curve
Standard geometry output.

Reverse Curve Node



The *Reverse Curve* node swaps the start and end of splines. The shape of the splines is not changed.

Tip

When used on the *Profile* input of the Curve to Mesh Node, this node fill flip the normals of the resulting mesh.

Inputs

Curve
: Standard geometry input.

Selection
: Whether or not to change the direction of each spline. True values mean the direction will be changed, false values mean the spline will be unaffected.

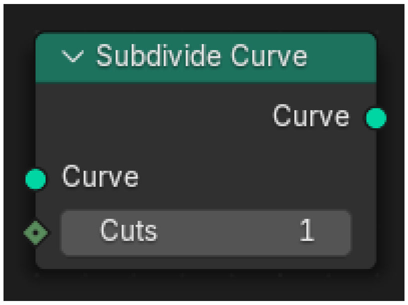
Properties

This node has no properties.

Outputs

Curve
: Standard geometry output.

Subdivide Curve Node



The *Subdivide Curve* node adds more control points in between existing control points on the curve input. For Bézier and poly splines, the shape of the spline will not be changed at all.

With Bézier curves, this can be used to increase the control on the shape of the curve while still having the higher-level provided by Bézier splines. Unlike the Resample Curve Node, where they are converted to poly splines.

Inputs

Curve
: Standard geometry input.

Cuts
: The number of control points to create on the segment following each point. When the input is a field, the number of cuts for a segment is determined by the value of the field when evaluated at the previous point.

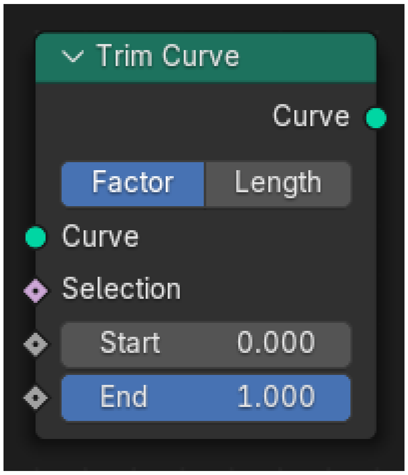
Properties

This node has no properties.

Outputs

Curve
Standard geometry output.

Trim Curve Node



The *Trim Curve* node shortens each spline in the curve by removing sections at the start and end of each spline.

Bézier splines will still be Bézier splines in the output, with the first and last control point and its handles moved as necessary to preserve the shape. NURBS splines will be transformed into poly splines in order to be trimmed.

Warning

Currently the Trim Curve node does not support cyclic splines.

Note

Since curve normals are calculated the final curve, this node may change the resulting normals when the **Minimum** twist method is used, since the **Minimum** method considers the entire length of the curve to decide the final normals. In some cases the Capture Attribute Node could be used to avoid this, by saving the original normals to be used later.

Inputs

Curve

Standard geometry input with a curve component.

Selection

A boolean field input for each curve indicating whether it is trimmed or not.

Start

The factor or length used to determine where to start each output spline.

> **Note**
>
> If the *Start* input is larger than the *End*, then the resulting spline will have a single point, located at the sample location of the *Start* value.

End

The factor or length used to determine where to end each output spline.

Properties

Mode

How to find endpoint positions for the trimmed spline.

Factor::	Find the endpoint positions using a factor of each spline's length. The input values should be between 0 or 1.
Length::	Find the endpoint positions using a length from the start of each spline. The input values should be between 0 and the length of the splines.

Outputs

Curve

Standard geometry output.

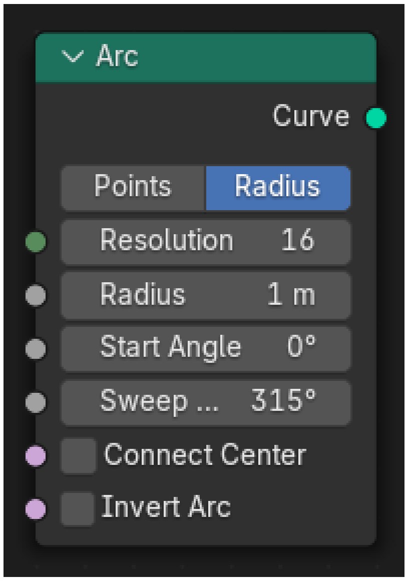
Curve Primitive Nodes

Nodes that create a primitive curve, e.g., a circle.

- Arc Node
- Bézier Segment Node
- Curve Circle Node
- Curve Line Node
- Curve Spiral Node
- Quadratic Bézier Node
- Quadrilateral Node
- Star Node

Arc Node

The *Arc* node generates a poly spline arc. The node has two modes, Radius and Points.



Resolution 16
Radius 1 m
Start Angle 0°
Sweep ... 315°
☐ Connect Center
☐ Invert Arc

Inputs

Resolution
: Number of edges on the arc.

Radius
: The radius of the arc. *Radius* mode only.

Start Angle
: Starting angle of the arc. *Radius* mode only.

Sweep Angle
: Length of the arc. *Radius* mode only.

Connect Center
: Connect the arc at the center.

Invert Arc
: Invert and draw opposite arc.

Start, Middle, End

The three points on the arc. *Points* mode only. The order of the points determines the direction (clockwise or counterclockwise) of the arc. The arc will always draw from Start to End via the Middle point. This can be changed by using the Invert Arc option.

Offset Angle

Offset angle of the arc. *Points* mode only.

Note

Because of the finite resolution, the middle point does not necessarily lie on the generated arc.

Properties

Mode

Points::	The position and radius of the arc are determined by three points. The center of the arc, radius and normal are also given as outputs.
Radius::	The arc is determined by the radius, start angle and sweep angle.

Outputs

Curve

Poly spline generated from the inputs.

Center

The center of the arc described by the three points. *Points* mode only.

Normal

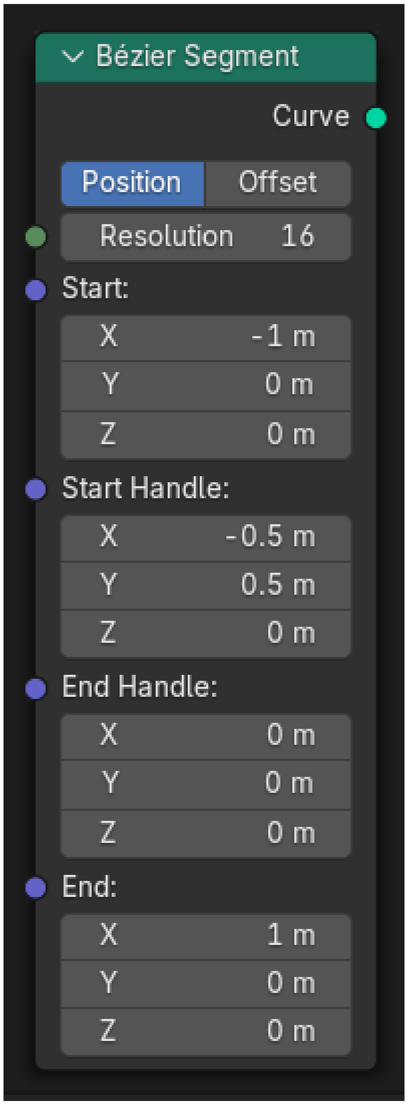
The normal direction of the plane described by the three points, pointing towards the positive Z axis. *Points* mode only.

Radius

The radius of the arc described by the three points. *Points* mode only.

Bézier Segment Node

The *Bézier Segment* node generates a 2D Bézier spline from the given control points and handles.


Bézier Segment
Curve
Position
Offset

Resolution 16

Start:
X -1 m
Y 0 m
Z 0 m

Start Handle:
X -0.5 m
Y 0.5 m
Z 0 m

End Handle:
X 0 m
Y 0 m
Z 0 m

End:
X 1 m
Y 0 m
Z 0 m

Inputs

Resolution
: The number of edges on the curve.

Start, End
: Positions of the start and end control point of the curve.

Start Handle, End Handle
Positions of the handles used to define the shape of the curve.

Properties

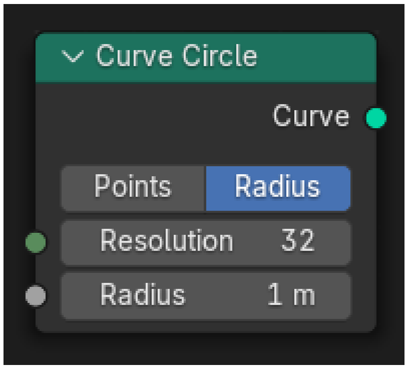
Mode

Position::	The handle inputs are the absolute positions of the handles in 3D space.
Offset::	The handle positions are relative to the control point on the curve. The handle inputs give the offset from the control points.

Outputs

Curve
Bézier spline generated from the inputs.

Curve Circle Node


Curve Circle
Curve
Points
Radius
Resolution 32
Radius 1 m

The *Curve Circle* node generates a poly spline circle.

Inputs

Resolution
: Number of edges on the circle.

Radius
: The radius of the circle.

Point 1, Point 2, Point 3
: The three points on the circle. The order of the points determines the direction (clockwise or counterclockwise) of the circle.

Note

Because of the finite resolution, the three points do not necessarily lie on the generated curve.

Properties

Mode

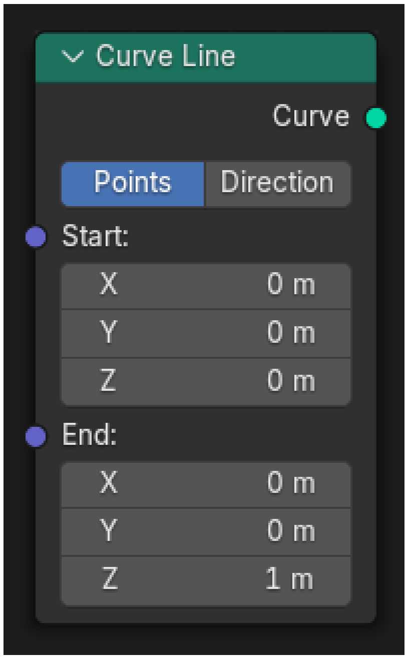
Points::	The position and radius of the circle are determined by three points. The center of the circle is also given as an output. If the three points lie on one line, no geometry is generated.
Radius::	The circle is determined by the radius.

Outputs

Curve
  Poly spline generated from the inputs.

Center
  The center of the circle defined by the three points.

Curve Line Node


Curve Line
Curve
Points
Direction
Start:
X
0 m
Y
0 m
Z
0 m
End:
X
0 m
Y
0 m
Z
1 m

The *Curve Line* node generates poly spline line.

Inputs

Start
: Position of the first control point.

End
: Position of the second control point. This is only available in the *Points* mode.

Direction
: Direction the line is going in. The length of this vector does not matter. This is only available in the *Direction* mode.

Length
: Length of the line. This is only available in the *Direction* mode.

Properties

Mode:

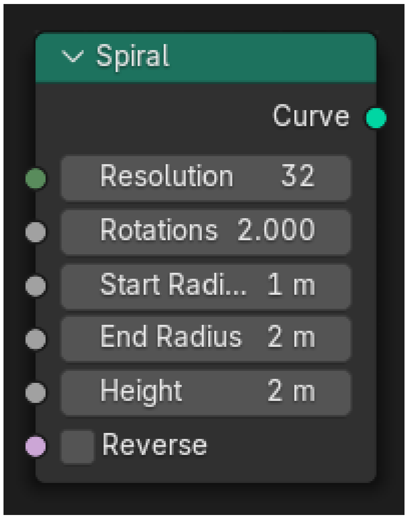
Points::	Define the spline with start and end points.
Direction::	Define the spline with a start, direction and length.

Outputs

Curve

Standard geometry output.

Curve Spiral Node


Spiral
Curve
Resolution 32
Rotations 2.000
Start Radi... 1 m
End Radius 2 m
Height 2 m
Reverse

The *Curve Spiral* node generates a poly spline in a spiral shape. It can be used to create springs or other similar objects. By default the spiral twists in a clockwise fashion.

Inputs

Resolution
: Number of points in one rotation of the spiral.

Rotations
: Number of times the spiral makes a full rotation.

Start Radius, End Radius
: Radius of the start point and end point of the spiral. The radius of the spiral changes linearly between the two values over the whole spiral.

Height
: Height of the spiral.

Reverse
: Boolean value that changes the direction from clockwise to counterclockwise when it is enabled.

Properties

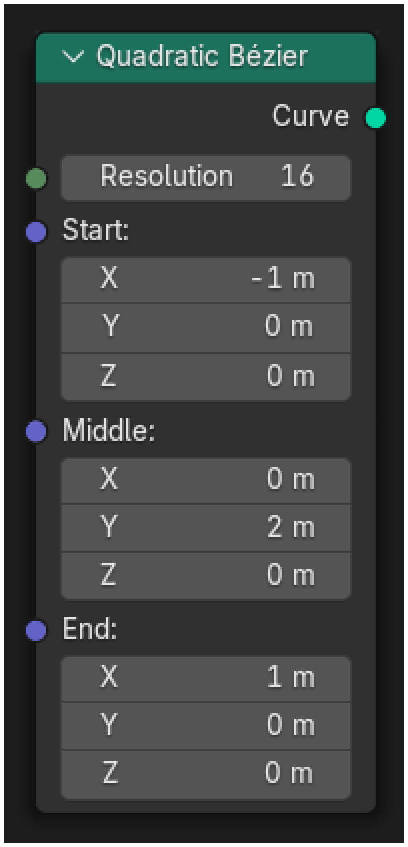
This node has no properties.

Outputs

Curve
: Poly spline generated from the inputs.

Quadratic Bézier Node

The *Quadratic Bézier* node generates a poly spline curve from the given control points. The generated shape is a parabola.



Resolution 16

Start:
X -1 m
Y 0 m
Z 0 m

Middle:
X 0 m
Y 2 m
Z 0 m

End:
X 1 m
Y 0 m
Z 0 m

Inputs

Resolution
: The number of edges on the curve.

Start, Middle, End
: Positions of the three control points. The generated curve passes through the two end points, and is tangent to the lines between the middle point and the two end points.

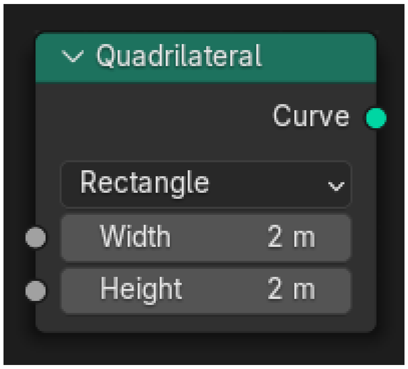
Properties

This node has no properties.

Outputs

Curve
    Poly spline generated from the inputs.

Quadrilateral Node



The *Quadrilateral* node generates a polygon with four points, with different modes.

Inputs

Width / Bottom Width / Top Width
: The X axis size of the shape.

Height
: The Y axis size of the shape.

Bottom Height / Top Height
: The distance between the bottom or top point and the X axis, in *Kite* mode.

Offset
: In *Parallelogram* mode, the relative X difference between the top and bottom edge. In *Trapezoid* mode, the amount to move the top edge in the positive X axis.

Point 1 - 4
: Input vectors for the *Points* mode.

Properties

Mode

Rectangle::	Generate a rectangle-shaped curve with a width and a height.
Parallelogram::	Generate a rectangle with an offset for the different X values of the top and bottom edges.
Trapezoid::	Generate a trapezoid-shaped curve with a height, a width for the top and bottom, and an offset.
Kite::	Generate a kite shape with a width, and the top and bottom points distance from the X axis.
Points::	Generate a four point cyclic poly spline by inputting the position vectors directly.

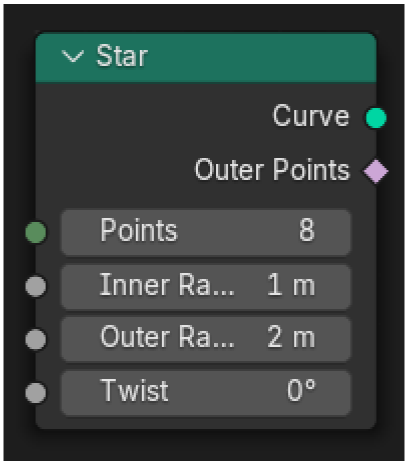
Outputs

Curve

Poly spline generated from the inputs.

Star Node

The *Star* node generates a poly spline in a star pattern by connecting alternating points of two circles. The points on the inner circle are offset by a rotation so that they lie in between the points on the outer circle. This offset can be changed with the twist input.



Points 8
Inner Ra... 1 m
Outer Ra... 2 m
Twist 0°

Inputs

Points
    Number of points on each of the circles.

Inner Radius, Outer Radius
    Radii of the two circles on which to place the control points. The inner radius can be larger than the outer radius.

Twist
    Angle offset of the inner circle. The twist value rotates the points on the circle corresponding with the inner radius counterclockwise by the given angle.

Properties

This node has no properties.

Outputs

Curve

Poly spline generated from the inputs.

Outer Points

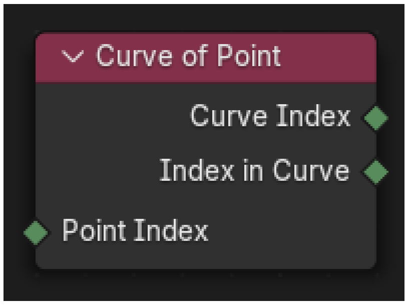
A boolean attribute field with a selection of the points on the *Outer Radius*, which is every other point.

Curve Topology Nodes

Nodes that retrieve information about the connectivity between curves and control points.

- Curve of Point Node
- Offset Point in Curve Node
- Points of Curve Node

Curve of Point Node



The *Curve of Point* node retrieves the index of the curve a control point is part of. This node is conceptually similar to the Face of Corner Node.

Inputs

Point Index
: The index of the input control point.

Note

By default this uses the index from the field context, which makes it important that the node is evaluated on the point domain.

Properties

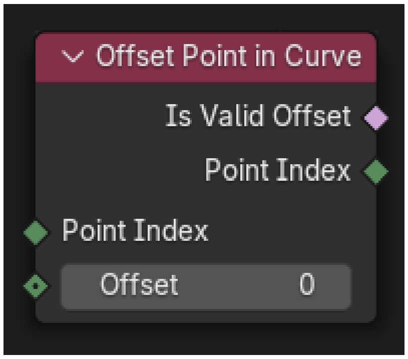
This node has no properties.

Outputs

Curve Index
: The index of the curve the control point is part of.

Index in Curve
: How far along the control point is along its curve, with a value of 0 for the first point in each curve.

Offset Point in Curve Node



The *Offset Point in Curve* node retrieves other points in the same curve as the input control point. This is like starting at a specific control point and walking along neighboring points toward the start or end of the curve.

Conceptually the operation is similar to the Offset Corner in Face Node, but the point index doesn't wrap around to the other end of the curve unless it is cyclic.

Inputs

Point Index

The index of the input control point.

Note

By default this uses the index from the field context, which makes it important that the node is evaluated on the point domain.

Offset

The number of points to move around the curve before finding the result. If the curve is cyclic and the offset goes past the start or end point of the curve, it will wrap around to the other side.

Properties

This node has no properties.

Outputs

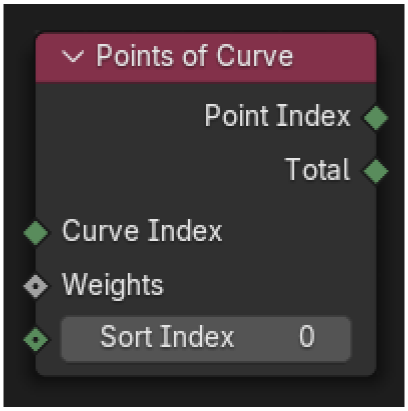
Is Valid Offset

Whether the input control point plus the offset is a valid index of the original curve. Any offset in a cyclic curve is always valid.

Point Index

The index of the offset curve point.

Points of Curve Node



The *Points of Curve* node retrieves indices of specific control points in a curve.

Inputs

Curve Index

The index of the input curve.

Note

By default this uses the index from the field context, which makes it important that the node is evaluated on the curve domain.

Weights

Values used to sort the curve's control points. By default the control points are sorted by index, so the control points with the smallest indices come first.

Sort Index

Which of the sorted control points to use for the *Point Index* output. If the value is larger than the total number of control points, it will wrap around to the beginning.

Properties

This node has no properties.

Outputs

Point Index
: The index of one of the curve’s control points, chosen by the *Sort Index* input.

Total
: The number of control points in the curve.

Instances Nodes

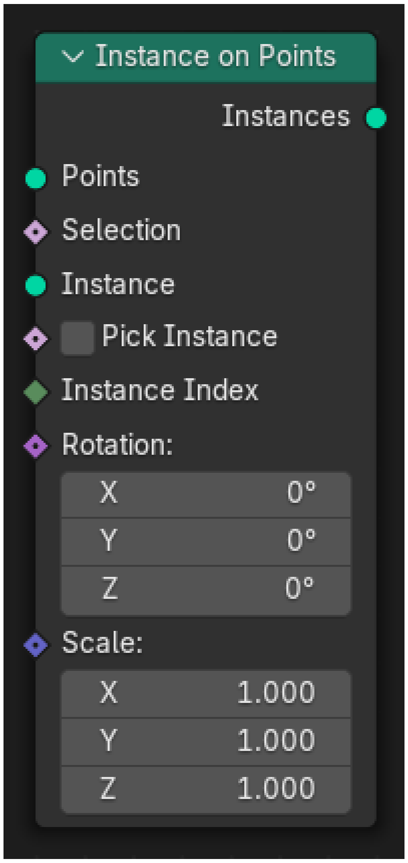
Nodes that only operate on instances.

- Instance on Points Node
- Instances to Points Node

- Realize Instances Node
- Rotate Instances Node
- Scale Instances Node
- Translate Instances Node
- Set Instance Transform Node

- Instance Transform Node
- Instance Rotation Node
- Instance Scale Node

Instance on Points Node


Instance on Points
Instances
Points
Selection
Instance
Pick Instance
Instance Index
Rotation:
X 0°
Y 0°
Z 0°
Scale:
X 1.000
Y 1.000
Z 1.000

The *Instance on Points* node adds a reference to a geometry to each of the points present in the input geometry. Instances are a fast way to add the same geometry to a scene many times without duplicating the underlying data. The node works on any geometry type with a *Point* domain, including meshes, point clouds, and curve control points.

Any attributes on the points from the *Geometry* input will be available on the instance domain of the generated instances.

Tip

The Make Instances Real operator can be used to create objects from instances generated with this node.

Note

To instance object types that do not contain geometry, like a light object, the Object Info Node can be used. Other objects like Metaball objects are not supported for instancing.

Inputs

Points

Standard geometry input. The position of the points of this geometry affect the transforms of each instance output.

> **Note**
>
> If the input geometry contains instances, the node will create more instances on the points inside the instances, creating nested instancing. In this case, each new instance will have the transform created by the node from the *Rotation* and *Scale* inputs, but it will *also* be transformed based on the parent instances.

Selection

Whether to instance on each point. True values mean an instance will be generated on the point, false values mean the point will be skipped.

Instance

The geometry to instance on each selected point. This can contain real geometry, or multiple instances, which can be useful when combined with the *Pick Instance* option.

Pick Instances

If enabled, instead of adding the entire geometry from the *Instance* input on every point, choose an instance from the instance list of the geometry based on the *Instance Index* input. This option is intended to be used with the Collection Info Node.

Instance Index

The selection of index for every selected point, only used when *Pick Instances* is true. By default the point ID is used, or the index if that doesn't exist. Negative values or values that are too large are wrapped around to the other end of the instance list.

Rotation

The Euler rotation for every instance. This can use the rotation output of nodes like Distribute Points on Faces and Curve to Points. An Euler rotation can also be created from a direction vector like the normal with the Align Rotation to Vector Node.

Scale

The size of each generated instance.

Properties

This node has no properties.

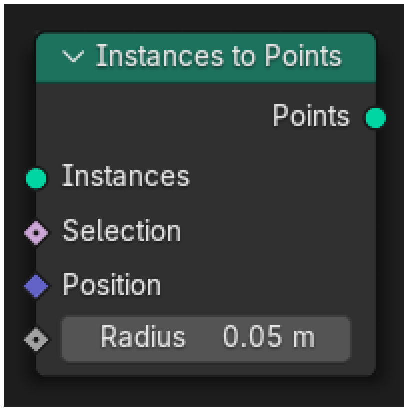
Outputs

Instances

Standard geometry output. If the `id` attribute exists on the input geometry, it will be copied to the result instances.

Instances to Points Node

The *Instances to Points* node generates points at the origins of top-level instances. Attributes on the instance domain are moved to the point cloud points.



Radius 0.05 m

Note

Top-level instances are those that are owned by the node’s input geometry. Instances owned by other instances, i.e. nested instances, are not considered by this node.

Inputs

Instances
: Standard geometry input.

Selection
: The instances used to generate points. True values mean a point is created for the instance, false values mean the instance is skipped.

Position
: Overrides the default position of generated point.

Radius
: Controls the radius of the result points.

Properties

This node has no properties.

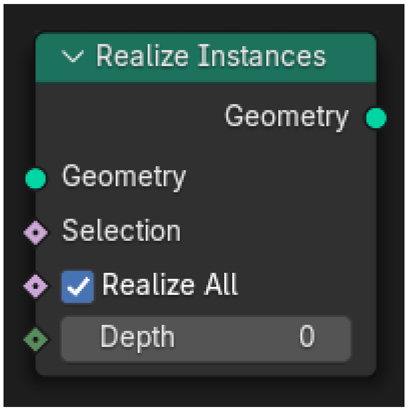
Outputs

Points

Standard geometry output.

Realize Instances Node

The *Realize Instances* node makes any instances (efficient duplicates of the same geometry) into real geometry data. This makes it possible to affect each instance individually, whereas without this node, the exact same changes are applied to every instance of the same geometry. However, performance can become much worse when the input contains many instances of complex geometry, which is a fundamental limitation when procedurally processing geometry.



☑ Realize All

Depth 0

Note

If the input contains multiple volume instances, only the first volume component is moved to the output.

Attributes

When merging attributes from multiple geometry inputs, the highest complexity data type is chosen for the output attribute. In other words, if a `weight` attribute has a Boolean type on one geometry input and a vector data type on another geometry, the `weight` attribute on the output geometry will have a vector data type.

Named and anonymous attributes are propagated from the instance domain to the realized geometry. If the same attribute exists on the geometry and on an instance, the attribute values from the geometry has precedence over the values on the instances.

In order to avoid creating duplicate values, the `id` attribute has special handling. The `id` values or indices of each instance are combined with id values from the points on geometry data.

Warning

Like other geometry nodes, this node always outputs generic typed attributes. So instead of a Vertex Group attribute, it will create a "Float" attribute on the result, and it will create a generic 2D vector attribute instead of a special "UV Map" attribute. Some other areas of Blender don't properly handle generic attributes in version 3.0.

Custom face corner normals are also not transferred currently.

Inputs

Geometry
: Standard geometry input.

Selection
: Which top-level instances to realize.

Realize All
: Realize all levels of nested instances for each top-level instances (overrides the value of the *Depth* input).

Depth
: Number of levels of nested instances to realize for each top-level instance.

Properties

This node has no properties.

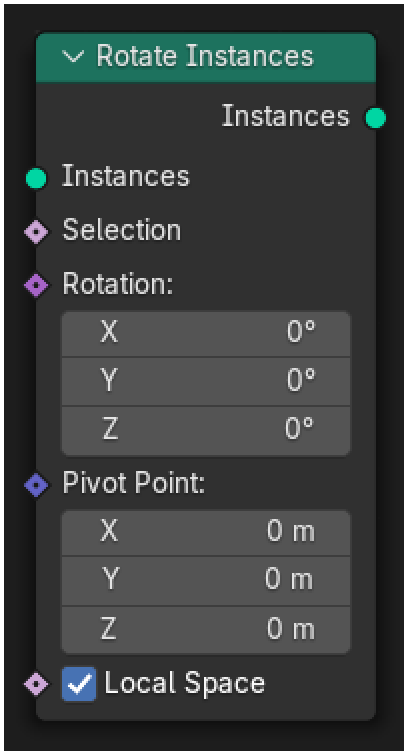
Outputs

Geometry
Standard geometry output.

Rotate Instances Node

The *Rotate Instances* node rotates geometry instances in local or global space.

The Instances page contains more information about geometry instances.


Rotate Instances
Instances
Instances
Selection
Rotation:
X 0°
Y 0°
Z 0°
Pivot Point:
X 0 m
Y 0 m
Z 0 m
☑ Local Space

Inputs

Instances
: Standard geometry input.

Selection
: Boolean field used to determine if an instance will be rotated.

Rotation
: The Euler rotation to rotate the instances by.

Pivot Point
: The position around which each instance is rotated. If the *Local Space* input is true, the location is relative to the initial transform of the instance.

Local Space
If enabled, the instances are rotated in local space. In other words, they are rotated around the axes described by the initial transform of each instance. When the input is disabled, the pivot point and rotation are specified in the local space of the modifier object.

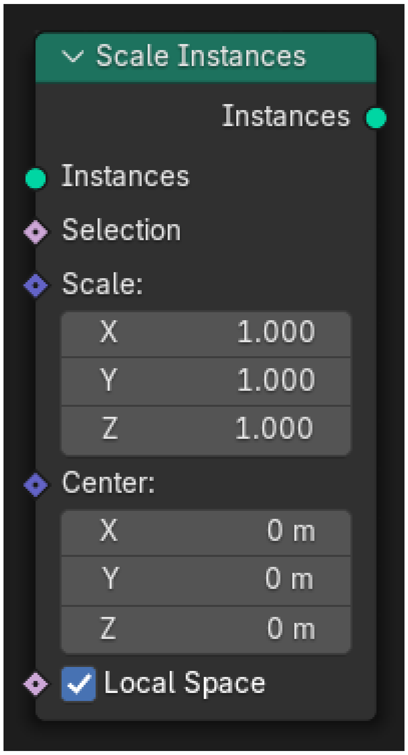
Properties

This node has no properties.

Outputs

Instances
Standard geometry output.

Scale Instances Node


Scale Instances
Instances
Instances
Selection
Scale:
X 1.000
Y 1.000
Z 1.000
Center:
X 0 m
Y 0 m
Z 0 m
Local Space

The *Scale Instances* node scales geometry instances in local or global space.

The Instances page contains more information about geometry instances.

Inputs

Instances
: Standard geometry input.

Selection
: Boolean field used to determine if an instance will be scaled.

Scale
: The scale factor to apply to the instance's transform on each axis.

Center
: The position from which the instance origins are scaled. Each instance will move away from this location. When the *Local Space* input is enabled, this location is relative to the initial transform of each instance.

Local Space
: If enabled, the instances are scaled in local space. In other words, they are scaled in the directions the described by the initial transform of each instance. When the input is disabled, the *Center* and *Scale* inputs are specified in the local space of the modifier object.

Properties

This node has no properties.

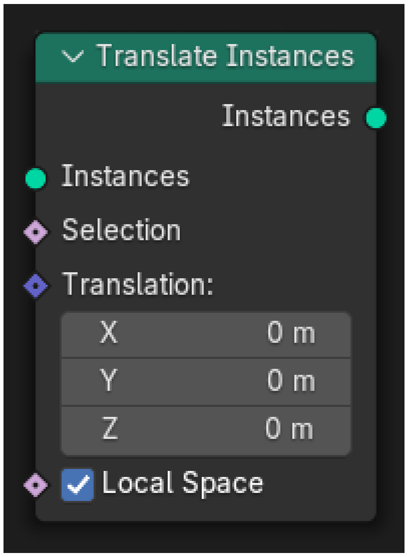
Outputs

Instances
: Standard geometry output.

Translate Instances Node

The *Translate Instances* node moves top-level geometry instances in local or global space.

The Instances page contains more information about geometry instances.


Translate Instances
Instances
Instances
Selection
Translation:

X 0 m
Y 0 m
Z 0 m
☑ Local Space

Inputs

Instances
: Standard geometry input.

Selection
: Boolean field used to determine if an instance will be translated.

Translation
: The vector to translate the instances by.

Local Space
: If enabled, the instances are translated relative to their initial rotation. Otherwise they are translated in the local space of the modifier object.

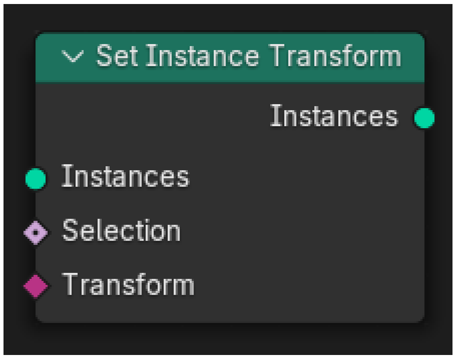
Properties

This node has no properties.

Outputs

Instances
    Standard geometry output.

Set Instance Transform Node


Set Instance Transform
Instances
Instances
Selection
Transform

The *Set Instance Transform* node Transforms geometry instances using a Transformation Matrix.

The Instances page contains more information about geometry instances.

Inputs

Instances
: Standard geometry input.

Selection
: Boolean field used to determine if an instance will be rotated.

Transform
: The transformation matrix to translate, rotate, and scale individual instances.

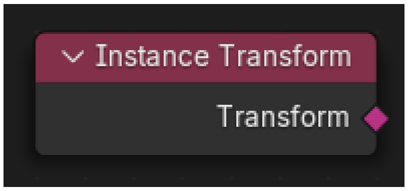
Properties

This node has no properties.

Outputs

Instances
: Standard geometry output.

Instance Transform Node



The *Instance Transform* outputs the Transformation Matrix of each top-level instance in the local space of the modifier object.

The Instances page contains more information about geometry instances.

Inputs

This node has no inputs.

Properties

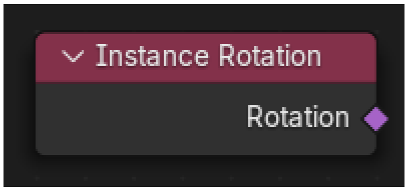
This node has no properties.

Outputs

Transformation

Matrix that indicates the Transformation of each top-level instance.

Instance Rotation Node



The *Instance Rotation* outputs the XYZ Euler rotation of each top-level instance in the local space of the modifier object.

The Instances page contains more information about geometry instances.

Note

Though rotations are often displayed in units of degrees in the spreadsheet or node editor, they are stored internally in radians, so this node outputs radians.

Inputs

This node has no inputs.

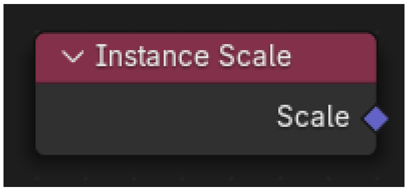
Properties

This node has no properties.

Outputs

Rotation
    Vector that indicates the rotation of each top-level instance in radians.

Instance Scale Node



The *Instance Scale* outputs the size of top-level instances on each axis in the local space of the modifier object.

The Instances page contains more information about geometry instances.

Inputs

This node has no inputs.

Properties

This node has no properties.

Outputs

Scale
    Vector that indicates the scale of each top-level instance.

Mesh Nodes

Nodes that only operate on meshes.

- Read
    - Edge Angle Node
    - Edge Neighbors Node
    - Edge Vertices Node
    - Edges to Face Groups Node
    - Face Area Node
    - Face Group Boundaries Node
    - Face Neighbors Node
    - Face Set Node
    - Is Face Planar Node
    - Is Edge Smooth Node
    - Is Face Smooth Node
    - Mesh Island Node
    - Shortest Edge Paths Node
    - Vertex Neighbors Node
- Sample
    - Sample Nearest Surface Node
    - Sample UV Surface Node
- Write
    - Set Face Set Node
    - Set Shade Smooth Node

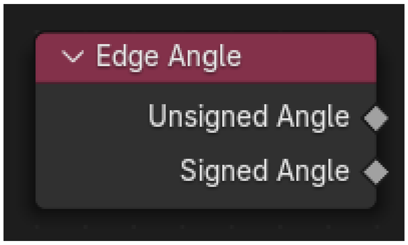
- Operations
    - Dual Mesh Node
    - Edge Paths to Curves Node
    - Edge Paths to Selection Node
    - Extrude Mesh Node
    - Flip Faces Node
    - Mesh Boolean Node
    - Mesh to Curve Node

  - Mesh to Points Node
  - Mesh to Volume Node
  - Scale Elements Node
  - Split Edges Node
  - Subdivide Mesh Node
  - Subdivision Surface Node
  - Triangulate Node
- Primitives
  - Cone Node
  - Cube Node
  - Cylinder Node
  - Grid Node
  - Icosphere Node
  - Mesh Circle Node
  - Mesh Line Node
  - UV Sphere Node
- Topology
  - Corners of Edge Node
  - Corners of Face Node
  - Corners of Vertex Node
  - Edges of Corner Node
  - Edges of Vertex Node
  - Face of Corner Node
  - Offset Corner in Face Node
  - Vertex of Corner Node
- UV
  - Pack UV Islands Node
  - UV Unwrap Node

Read Mesh Nodes

- Edge Angle Node
- Edge Neighbors Node
- Edge Vertices Node
- Edges to Face Groups Node
- Face Area Node
- Face Group Boundaries Node
- Face Neighbors Node
- Face Set Node
- Is Face Planar Node
- Is Edge Smooth Node
- Is Face Smooth Node
- Mesh Island Node
- Shortest Edge Paths Node
- Vertex Neighbors Node

Edge Angle Node



The *Edge Angle* node calculates the angle in radians between two faces that meet at an edge. For the Face, Face Corner, and Point domains, the node uses simple domain interpolation to move values from the mesh’s edges.

Note

The output of this node depends on the density of the mesh. If there are more edges closer together and the curvature of the mesh stays the same, the edge angle will be different

Inputs

This node has no inputs.

Properties

This node has no properties.

Outputs

Unsigned Angle

The shortest angle in radians between two faces where they meet at an edge. The range of the data is from zero to PI. Flat edges and Non-manifold edges have an angle of zero. An edge between two faces completely folded back on each other has an angle of PI, or 180 degrees.

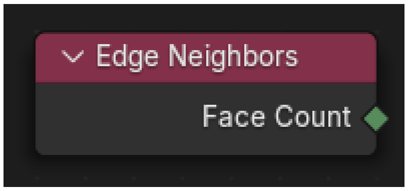
Tip

Computing this value is slightly faster than the signed angle, so if there is no need to distinguish between convex and concave angles, using this value can provide a performance improvement.

Signed Angle

The signed angle in radians between two faces where they meet at an edge. Flat edges and Non-manifold edges have an angle of zero. Concave angles are positive and convex angles are negative.

Edge Neighbors Node



The *Edge Neighbors* node outputs topology information relating to each edge of a mesh.

Inputs

This node has no inputs.

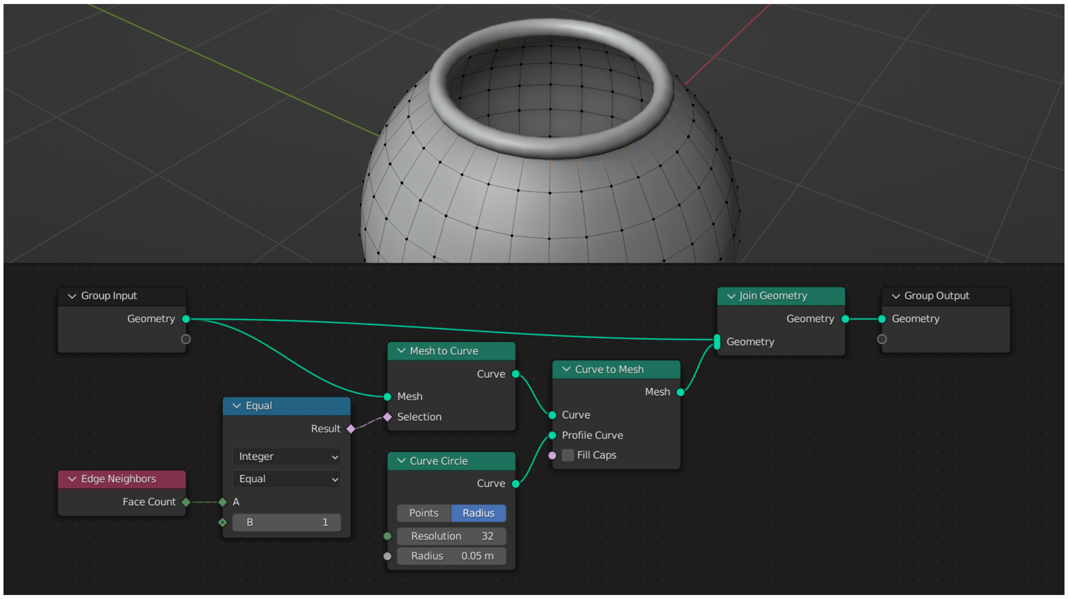
Properties

This node has no properties.

Outputs

Face Count
: The number of faces that use the edge as one of their sides. When the value is one, the edge is a non-manifold boundary edge. Alternatively, when the value is zero, the edge is a loose edge, not used by any faces.

Examples


Group Input
Geometry
Mesh to Curve
Curve
Mesh
Selection
Equal
Result
A
Edge Neighbors
Face Count
Curve to Mesh
Mesh
Curve
Profile Curve
Fill Caps
Curve Circle
Curve
Points
Radius
Join Geometry
Geometry
Geometry
Group Output
Geometry

Integer

Equal

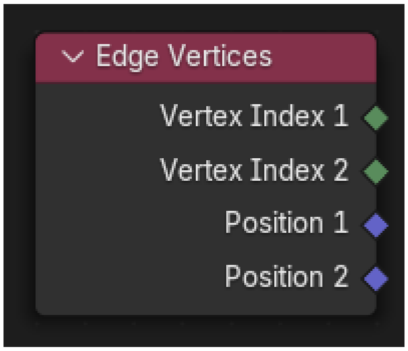
B 1

Resolution 32

Radius 0.05 m

Using the *Face Count* output to create a curve on a mesh's boundary edges.

Edge Vertices Node



The *Edge Vertices* node outputs the position and index of the two vertices of each of a mesh’s edges.

Note

The order of the two vertices of an edge is arbitrary. In some cases it may be predictable based on the internals of the algorithm that created the mesh, but in general the order should not be relied upon.

Inputs

This node has no inputs.

Properties

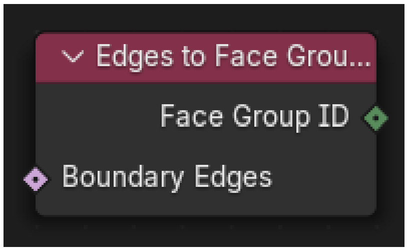
This node has no properties.

Outputs

Vertex Index 1/2
The index of the two vertices of the edge.

Position 1/2
The position of each of the edge’s vertices. This output is for convenience, it is the same as using index output to retrieve the position from the Evaluate at Index Node.

Edges to Face Groups Node


Edges to Face Grou...
Face Group ID
Boundary Edges

The *Edges to Face Groups* node group faces into regions surrounded by the selected boundary edges.

Inputs

Boundary Edges
: Edges used to split faces into separate groups.

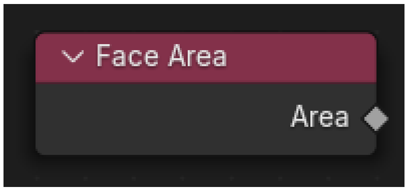
Properties

This node has no properties.

Outputs

Face Group ID
: Index of the face group inside each boundary edge region.

Face Area Node



The *Face Area* node outputs the surface area of a mesh’s faces. The units are in Blender units no matter the unit system, equivalent to meters-squared at the default unit scale.

Note

For quads and N-gons, when the face’s vertices are not planar, the output is not necessarily the same as the sum of every one of the face’s triangles visible in the viewport. In this case it should only be used an approximation. In some cases, the Triangulate Node can be used to get an exact value.

Inputs

This node has no inputs.

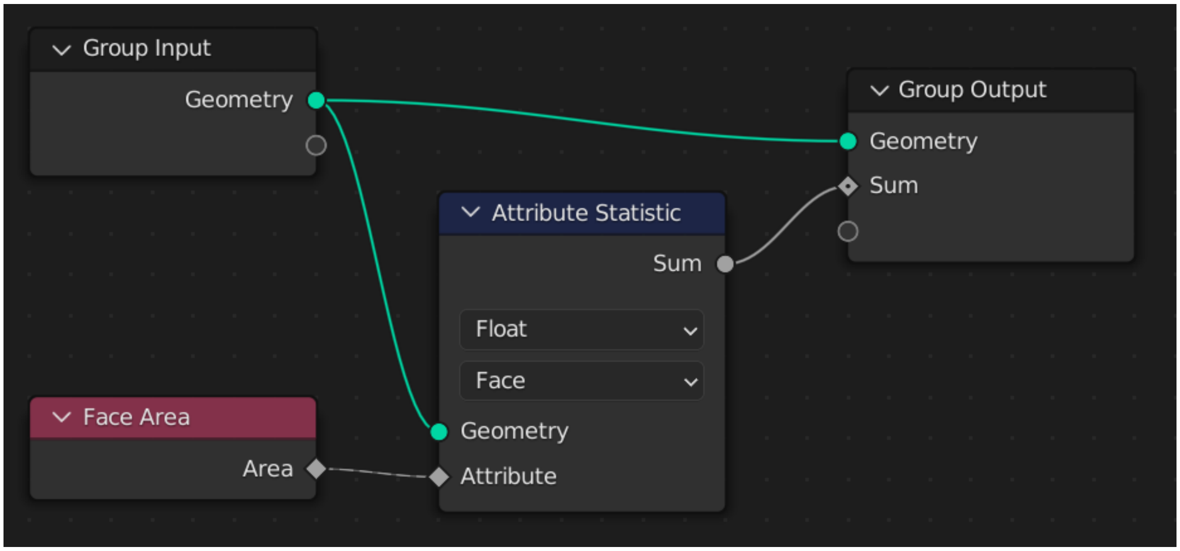
Properties

This node has no properties.

Outputs

Area
    The surface area of each of the mesh’s faces.

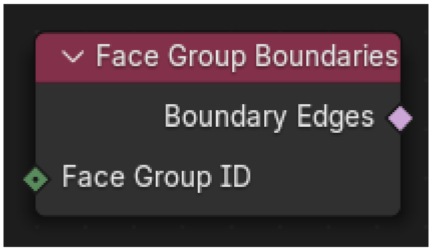
Examples


Group Input
Geometry
Face Area
Area
Attribute Statistic
Sum
Float
Face
Geometry
Attribute
Group Output
Geometry
Sum

Combined with the Attribute Statistic Node, this node can be used to calculate the total surface area of a mesh.

Face Group Boundaries Node

The *Face Group Boundaries Node* finds the edges which lie on the boundaries of specified regions. These edges could be used to mark seams for UV unwrapping, for example.



Inputs

Face Group ID

Identifier for which group of faces this face belongs to. All contiguous faces with the same value are in the same region.

Properties

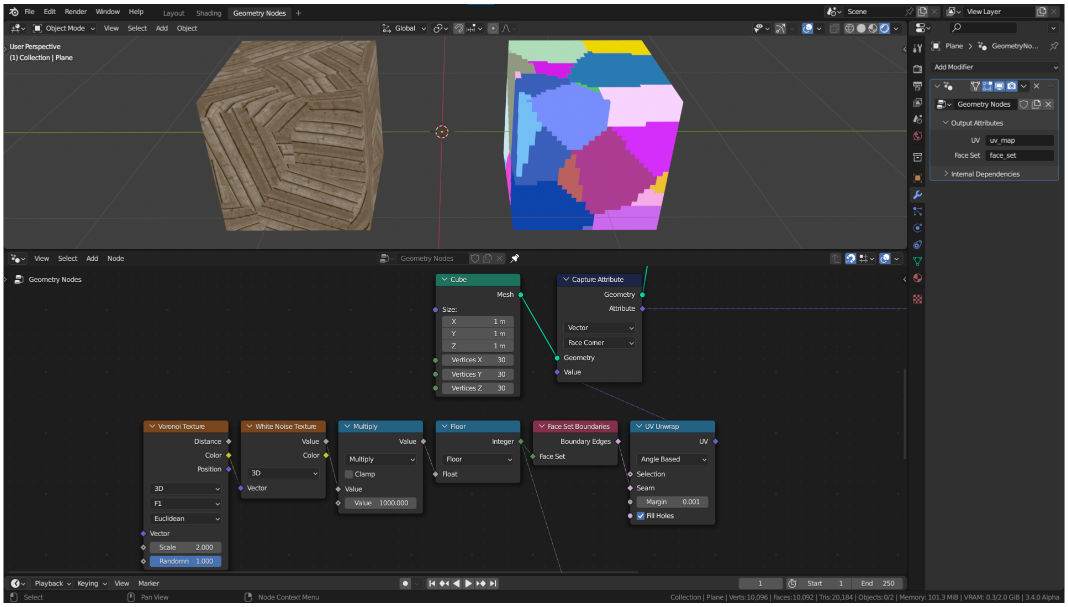
This node has no properties.

Output

Boundary Edges

Selection of the boundary edges of the different face sets. An edge is considered to be at the boundary if it lies on at least two faces with different identifiers.

Examples


Cube
Mesh
Size:
X 1 m
Y 1 m
Z 1 m
Vertices X 30
Vertices Y 30
Vertices Z 30
Capture Attribute
Geometry
Attribute
Vector
Face Corner
Geometry
Value
Voronoi Texture
Distance
Color
Position
3D
F1
Euclidean
Vector
Scale 2.000
Randomn 1.000
White Noise Texture
Value
Color
3D
Vector
Multiply
Value
Multiply
Clamp
Value
Value 1000.000
Floor
Integer
Floor
Float
Face Set Boundaries
Boundary Edges
Face Set
UV Unwrap
UV
Angle Based
Selection
Seam
Margin 0.001
Fill Holes

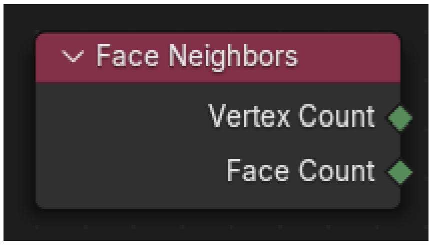
UV uv_map

Face Set face_set

Combined with the UV Unwrap Node, this node is used to turn the face sets (right cube) into a UV map for a texture (left cube).

Face Neighbors Node

The *Face Neighbors* node outputs topology information relating to each face of a mesh.



Inputs

This node has no inputs.

Properties

This node has no properties.

Outputs

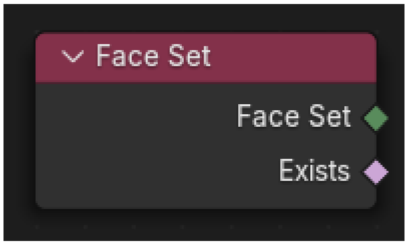
Vertex Count

This output is simply the number of sides of each face, or how many corners each face has.

Neighboring Face Count

The number of faces that connect to this face with at least one edge. On a regular manifold mesh with only quads and triangles, this will be the same as the vertex count, otherwise it might be completely different.

Face Set Node



The *Face Set Node* outputs which face set a face is in, and whether or not face sets exist in the mesh at all.

The corresponding data flow node is the Set Face Set Node.

Note

This node can only be used in the Tool context.

Inputs

This node has no inputs.

Properties

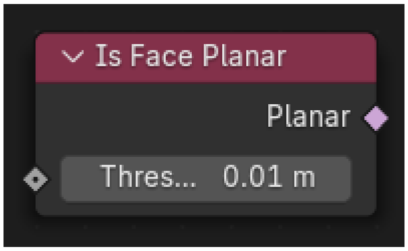
This node has no properties.

Output

Face Set
: Integer indicating which face set a face is in, or 0 when the mesh does not have face sets. When evaluated in the edge or point domain, outputs an interpolated value based on the connected faces.

Exists
: Boolean value that indicates whether the element’s mesh has face sets.

Is Face Planar Node



The *Is Face Planar* node outputs whether every triangle of a quads or N-gons is on the same plane as all of the others, in other words, if they have the same normal.

For example, a non-planar face can be created by moving a single vertex in a face but not the others. Triangles will always be planar.

Inputs

Threshold
: The distance a point can be from the surface before the face is no longer considered planar.

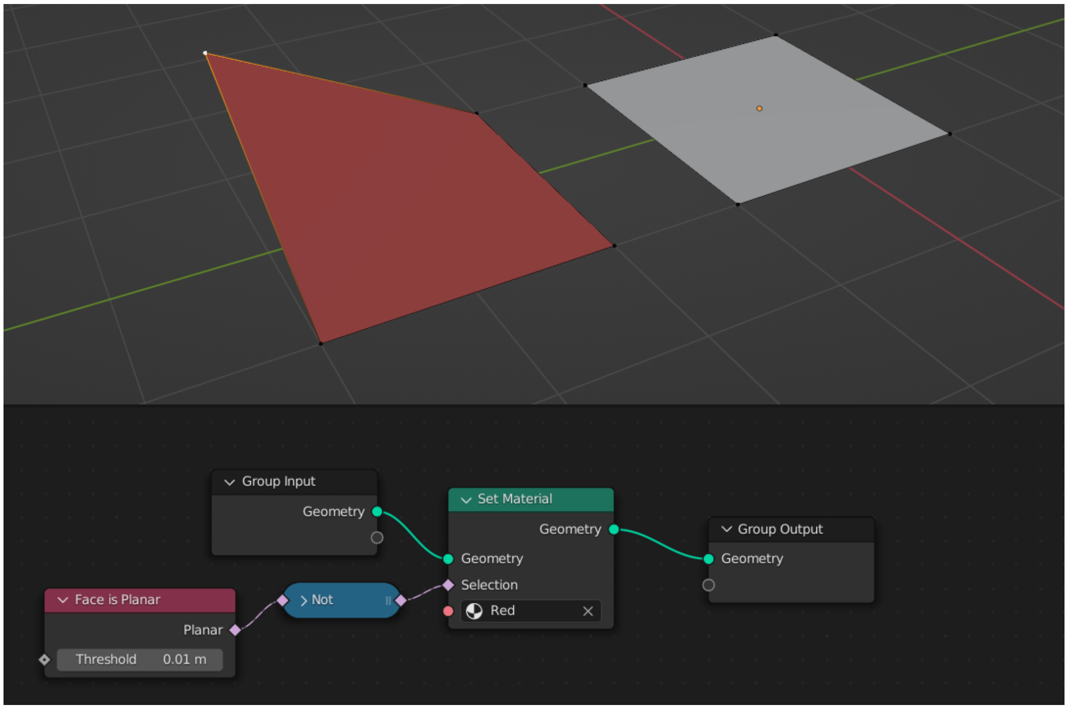
Properties

This node has no properties.

Outputs

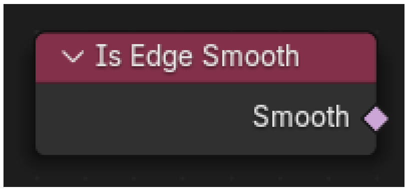
Planar
: Whether each mesh face is planar.

Examples


Group Input
Geometry
Set Material
Geometry
Geometry
Selection
Red
Group Output
Geometry
Face is Planar
Planar
Threshold 0.01 m
Not

Combined with the Set Material Node, this node is used to visualize all non-planar faces in a mesh.

Is Edge Smooth Node



The *Is Edge Smooth* node outputs true for each edge of the mesh that is *not* marked as sharp. Otherwise, if the edge *is* marked as sharp, then the node outputs false.

See also

Mark Sharp & Clear Sharp

Inputs

This node has no inputs.

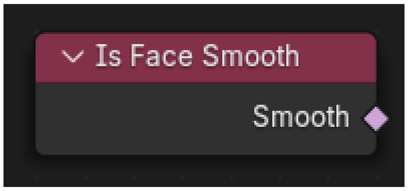
Properties

This node has no properties.

Outputs

Smooth
  Boolean value that indicates whether the edges of the mesh are *not* marked as sharp.

Is Face Smooth Node



The *Is Face Smooth* node outputs true for each face of the mesh if that face is marked to render smooth shaded. Otherwise, if the face is marked to render as flat shaded, then the node outputs false.

Inputs

This node has no inputs.

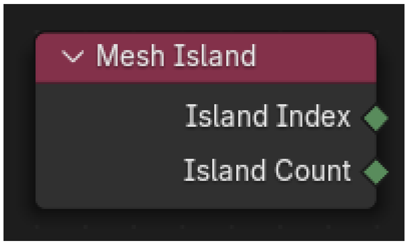
Properties

This node has no properties.

Outputs

Smooth
: Boolean value that indicates whether the normals of each face corner on the final mesh are smoothed with normal of all adjacent faces or not.

Mesh Island Node



The *Mesh Island* node outputs information about separate connected regions, or “islands” of a mesh. Whenever two vertices are connected together by an edge, they are considered as part of the same island, and will have the same *Island Index* output.

This node’s behavior is similar to the Select Linked operator in edit mode, or the *Random per Island* output of the Geometry shader node.

Inputs

This node has no inputs.

Properties

This node has no properties.

Outputs

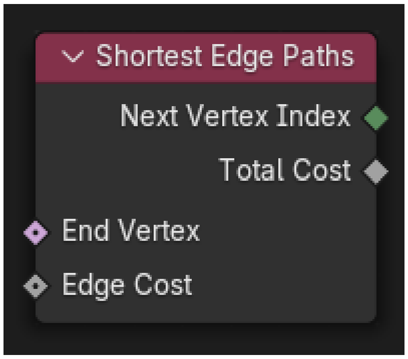
Island Index

> The index of each vertex’s island. Indices are decided based on the lowest vertex index contained in each island.

Island Count

> The total number of mesh islands. This is a single value, and does not vary per element.

Shortest Edge Paths Node

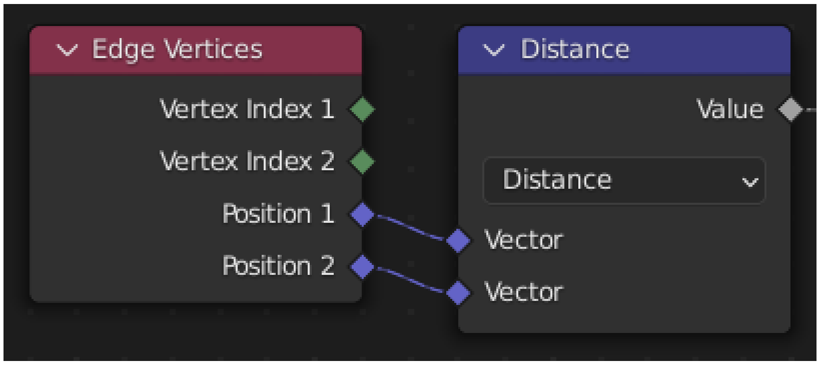

Shortest Edge Paths
Next Vertex Index
Total Cost
End Vertex
Edge Cost

The *Shortest Edge Paths* node finds paths along mesh edges to a selection of end vertices. The cost used to define “shortest” can be set to anything. By default there is a constant cost for every edge, but a typical input would be the length of each edge.

The output is encoded with vertex indices, and is meant to be used on the vertex domain. For each vertex, the *Next Vertex Index* output gives the index of the following vertex in the path to the “closest” endpoint.

The node is implemented with Dijkstra’s algorithm.

Tip


Edge Vertices
Vertex Index 1
Vertex Index 2
Position 1
Position 2
Distance
Value
Distance
Vector
Vector

The edge length is a natural input to the *Edge Cost*. It can be implemented with the Edge Vertices Node and the Vector Math Node set to the *Distance* operation.

See also

This node can be used with the Edge Paths to Selection Node or the Edge Paths to Curves Node to generate new geometry based on the paths.

Inputs

End Vertex
: A selection of the goal vertices that terminate the edge paths.

Edge Cost
: The weight for each edge, used to determine the meaning of “shortest.”

Properties

This node has no properties.

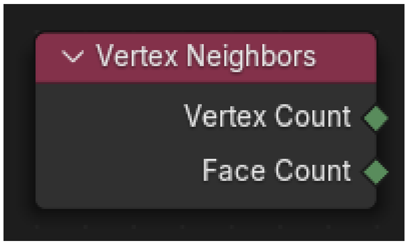
Outputs

Next Vertex Index
: The following vertex on the shortest path from every vertex to the closest endpoint (as defined by the cost input).

Total Cost
: The remaining cost before an end vertex is reached.

Vertex Neighbors Node

The *Vertex Neighbors* node outputs topology information relating to each vertex of a mesh.


Vertex Neighbors
Vertex Count
Face Count

Inputs

This node has no inputs.

Properties

This node has no properties.

Outputs

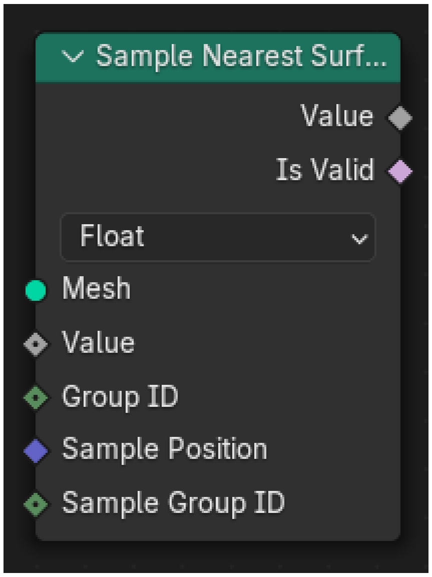
Vertex Count
    The number of vertices connected to this vertex with an edge, equal to the number of connected edges.

Face Count
    The number of faces that contain the vertex.

Mesh Sample Nodes

- Sample Nearest Surface Node
- Sample UV Surface Node

Sample Nearest Surface Node


Sample Nearest Surf...
Value
Is Valid
Float
Mesh
Value
Group ID
Sample Position
Sample Group ID

The *Sample Nearest Surface* node finds values at the closest points on the surface of a source mesh geometry. Non-face attributes are interpolated across the surface.

This node is similar to the Geometry Proximity Node, but it gives the value of any attribute at the closest surface point, not just its position.

Warning

Because the node samples the *surface* of a mesh rather than its edges or vertices, values from loose points and edges are ignored.

Inputs

Mesh
: The geometry to retrieve the attribute from.

Value

A field to evaluate on the *Source* geometry for use with the transfer method.

Group ID
Is evaluated on the face domain and splits the input mesh into multiple parts, each with its own id.

Sample Position
The position to start from when finding the closest location on the target mesh. By default, this is the same as if the Position Node was connected.

Sample Group ID
Determines in which group the closest nearest surface is detected.

Properties

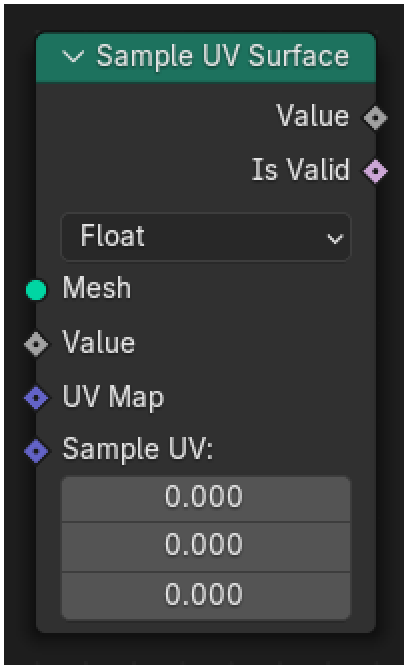
Data Type
The data type to use for the retrieved values.

Outputs

Value
The data retrieved and interpolated from the *Source* geometry, mapped based on the node’s settings and inputs.

Is Valid
Whether the sampling was successful. It can fail when the sampled group is empty.

Sample UV Surface Node


Sample UV Surface
Value
Is Valid
Float
Mesh
Value
UV Map
Sample UV:
0.000
0.000
0.000

The *Sample UV Surface* node finds values on a mesh’s surface at specific UV locations. Internally the process is a “reverse UV lookup” from a location in 2D space. The node then finds the face that corresponds to each UV coordinate, and the location within that face.

Warning

Because of the node’s method of computation, the UV map should not have any overlapping faces. If the UV map is sampled at a location with no faces or overlapping faces, the node will output the default value for the data type, which is zeros for most types.

Inputs

Mesh

A geometry containing the mesh with a UV map for sampling.

Value
A field to evaluate on the target *Mesh* geometry for later sampling at the surface positions.

UV Map
The mesh UV map to sample, evaluated on the *Mesh* input. Should not have overlapping faces.

Sample UV
The coordinates to sample within the UV map.

Properties

Data Type
The data type to use for the retrieved values.

Outputs

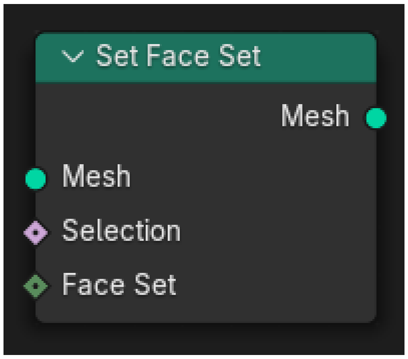
Value
The data retrieved and interpolated from the *Mesh* geometry, mapped based on the node's settings and inputs.

Is Valid
Whether the node could find a single face to sample at the UV coordinate.

Write Mesh Nodes

- Set Face Set Node
- Set Shade Smooth Node

Set Face Set Node



The *Set Face Set* node controls which face set that faces are in.

The input node for this data is the Face Set Node.

Note

This node can only be used in the Tool context.

Inputs

Mesh
: Standard geometry input.

Selection
: Boolean field that controls which faces will have the Face Set value applied.

Face Set
: Integer field for specifying which face set each selected face should be moved to. Ignored for faces where the value of Selection is false.

Properties

This node has no properties.

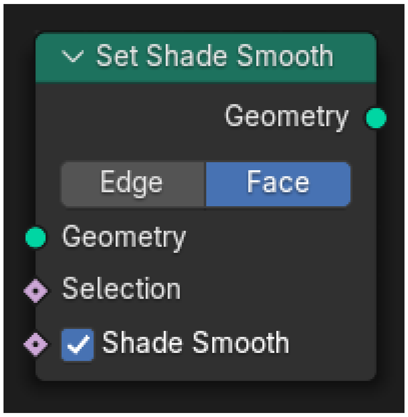
Outputs

Mesh

Standard geometry output.

Set Shade Smooth Node

The *Set Shade Smooth* node controls whether the mesh's faces look smooth in the viewport and renders. The smooth status of both edges and faces can be controlled, corresponding to the `sharp_edge` and `sharp_face` attributes. The input node for this data is the Is Face Smooth Node.



Inputs

Mesh
: Standard geometry input.

Shade Smooth
: When true, the selected faces will be marked to render smooth shaded. Otherwise the faces will be rendered flat shaded.

Selection
: Boolean input for selecting which faces will have the Shade Smooth value applied.

Properties

Domain
: Whether to write smoothness of mesh faces or edges.

Outputs

Mesh

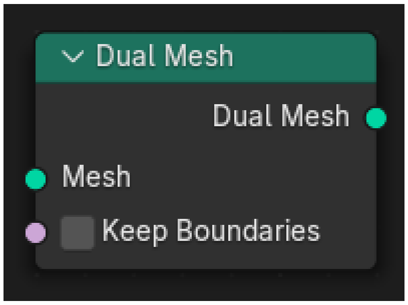
Standard geometry output.

Mesh Operation Nodes

Nodes that only operate on meshes.

- Dual Mesh Node
- Edge Paths to Curves Node
- Edge Paths to Selection Node
- Extrude Mesh Node
- Flip Faces Node
- Mesh Boolean Node
- Mesh to Curve Node
- Mesh to Points Node
- Mesh to Volume Node
- Scale Elements Node
- Split Edges Node
- Subdivide Mesh Node
- Subdivision Surface Node
- Triangulate Node

Dual Mesh Node



The *Dual Mesh Node* converts a mesh into its dual, i.e. faces are turned into vertices and vertices are turned into faces. This also means that attributes which were on the face domain are transferred to the point domain in the dual mesh.

Warning

The Dual Mesh node only works on manifold geometry. To work with non-manifold geometry it's best to remesh the geometry first.

Inputs

Mesh
  Standard geometry input.

Keep Boundaries
  Keeps the non-manifold boundaries of the input mesh in place, by creating extra geometry, and avoiding the dual mesh transformation there.

Properties

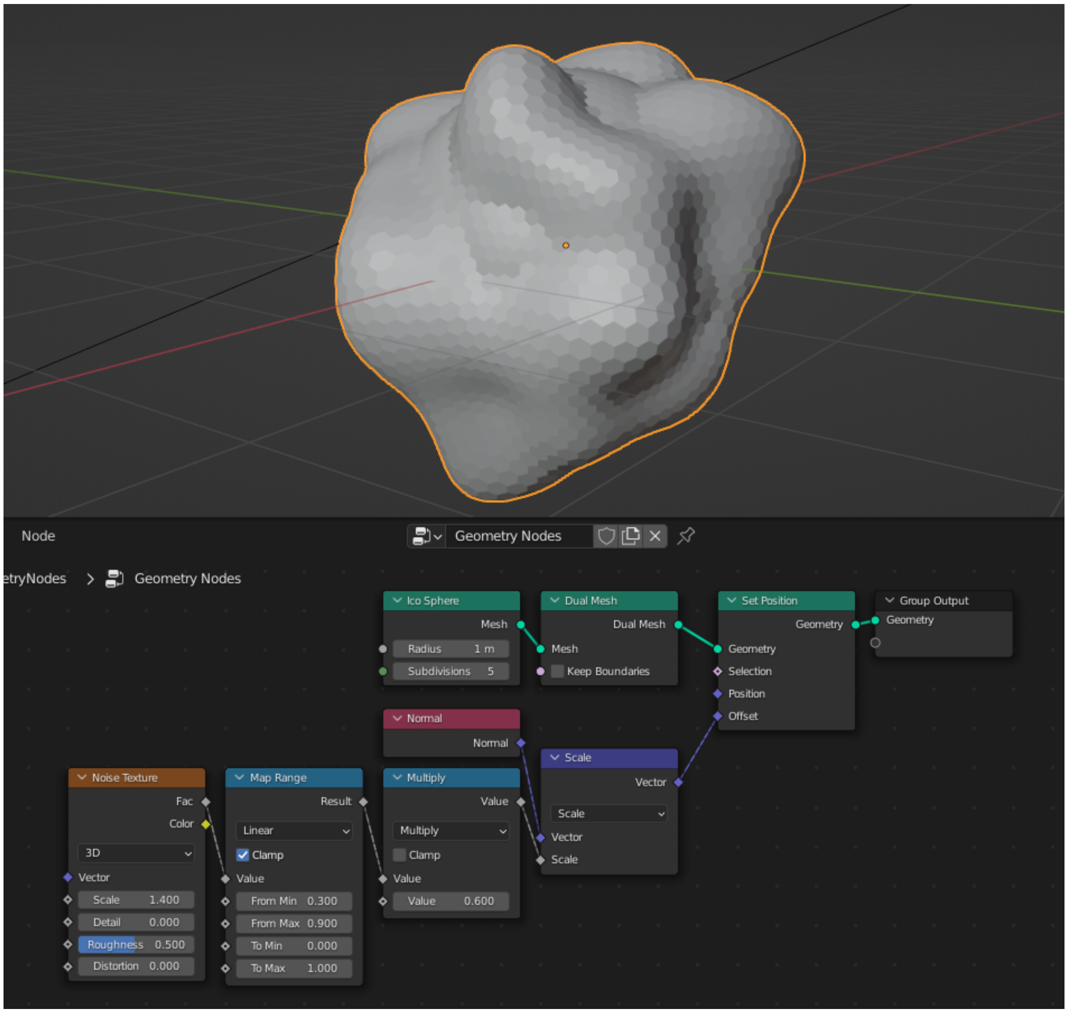
This node has no properties.

Output

Dual Mesh
  Standard geometry output.

Examples

The *Dual Mesh Node* combines nicely with triangulated meshes. In this case an Ico Sphere is used, which is made up of nice and evenly spaced triangles.


Node
Geometry Nodes
etryNodes > Geometry Nodes
Ico Sphere
Mesh
Dual Mesh
Dual Mesh
Mesh
Set Position
Geometry
Geometry
Selection
Position
Offset
Group Output
Geometry
Normal
Normal
Scale
Vector
Vector
Scale
Noise Texture
Fac
Color
Vector
Map Range
Result
Value
Multiply
Value
Value

Ico Sphere: Radius 1 m, Subdivisions 5

Dual Mesh: ☐ Keep Boundaries

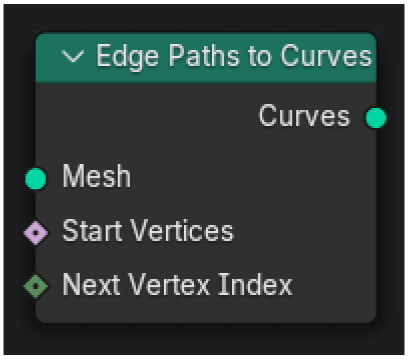
Scale: Scale

Noise Texture: 3D, Scale 1.400, Detail 0.000, Roughness 0.500, Distortion 0.000

Map Range: Linear, ☑ Clamp, From Min 0.300, From Max 0.900, To Min 0.000, To Max 1.000

Multiply: Multiply, ☐ Clamp, Value 0.600

Edge Paths to Curves Node


Edge Paths to Curves
Curves
Mesh
Start Vertices
Next Vertex Index

The *Edge Paths to Curves* node output curves that follow paths across mesh edges.

See also

This node is meant to use the output of the Shortest Edge Paths Node. It is similar to the Edge Paths to Selection Node, but it creates a curve that follow each path, rather than a selection of every visited edge.

Inputs

Mesh
: Standard mesh input.

Start Vertices
: A selection of the vertices to start at when traveling along the next vertex indices.

Next Vertex Index
: Describes the path to follow at every vertex.

Properties

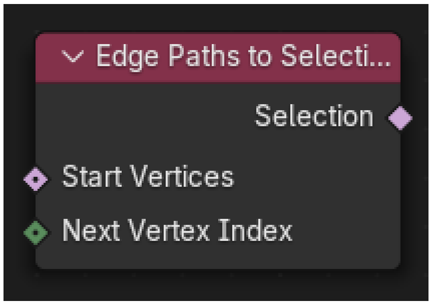
This node has no properties.

Outputs

Mesh

Standard curves output.

Edge Paths to Selection Node


Edge Paths to Selecti...
Selection
Start Vertices
Next Vertex Index

The *Edge Paths to Selection* node follows paths across mesh edges and outputs a selection of every visited edge.

See also

This node is meant to use the output of the Shortest Edge Paths Node. It can be combined with the Separate Geometry Node to remove any unused edges.

Inputs

Start Vertices
: A selection of the vertices to start at when traveling along the next vertex indices.

Next Vertex Index
: Describes the path to follow at every vertex.

Properties

This node has no properties.

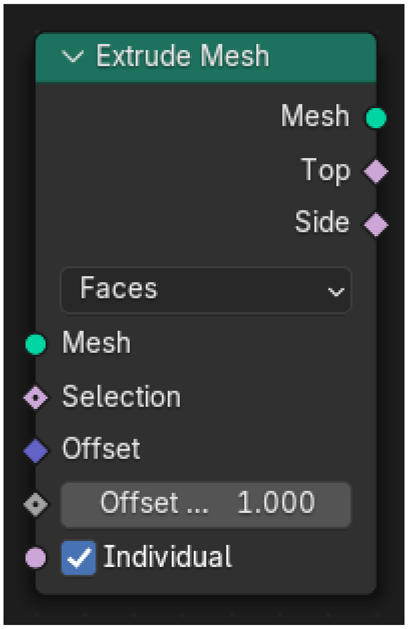
Outputs

Selection
: A boolean field indicating all edges visited when traversing the mesh.

Extrude Mesh Node

The *Extrude Mesh Node* generates new edges or faces on the selected geometry elements and moves them by a certain offset.

The operations are similar to the extrude tools in mesh edit mode, though there are some differences. Most importantly, the node never keeps the back-faces of the extrusion in place, they are always removed. Attribute propagation rules may also be different.



Inputs

Mesh

Standard geometry input.

Selection

A boolean field indicating which elements should be extruded.

Offset

The translation vector for each extruded element. By default, this is the element’s normal.

> **Tip**
>
> If all the elements are extruded in the same direction, you may be able to improve performance by connecting a Vector Node to this input, thereby skipping the normal calculation.

Offset Scale

Scaling factor for the above translation vector.

Individual *Face Mode Only*

Whether to extrude each face individually rather than extruding connected groups of faces together.

Properties

Mode

Vertices::	Attaches a new freestanding edge to each selected vertex.
Edges::	Attaches a new quad face to each selected edges. Vertices shared by the original selected edges are also shared in the duplicated edges. **Note** Depending on the situation, the normals of the new faces may be arbitrary. If the selected edges each have only one connected face, then the node can pick a consistent orientation for the new faces, but if there is more than one connected face, or no connected faces, the normals may have to be adjusted afterwards.
Faces::	Extrudes contiguous regions of selected faces, or each selected face individually, depending on the *Individual* boolean input. When the *Individual* input is false, the node will find regions of connected faces and generate new “side” faces on the boundaries of those regions. Any vertices, edges or faces on the *inside* of the regions simply are moved, not duplicated. If the whole mesh is selected and it is already a Manifold shape, then the result will just be that the whole mesh gets resized.

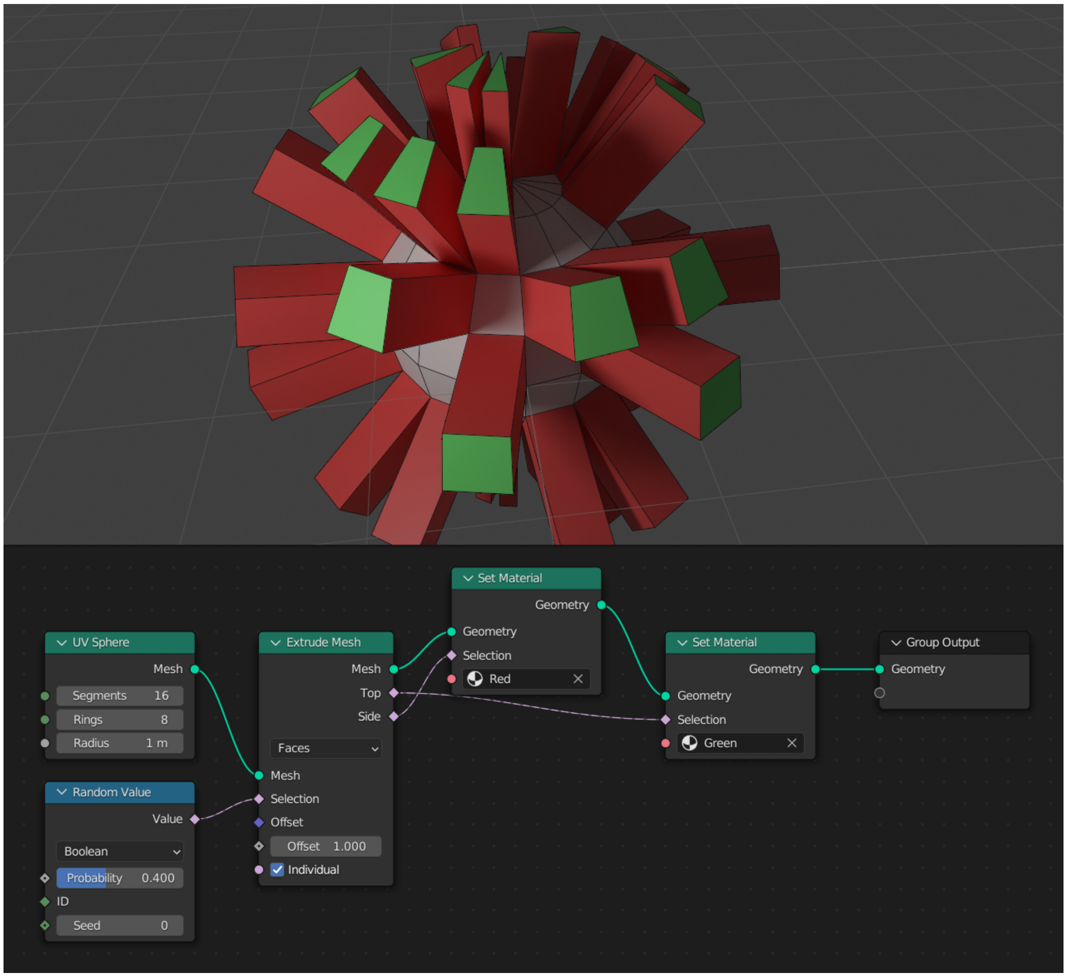
Output

Mesh
: Standard geometry output.

Top
: A boolean field indicating the “top” elements in the extrusion. In *Vertex* mode, these are the new vertices; in *Edge* mode, the new edges; and in *Face* mode, the moved faces.

Side
: A boolean field indicating the “side” elements in the extrusion. In *Vertex* mode, these are the new edges; in *Edge* mode, the new faces; and in *Face* mode, too, the newly generated faces (as opposed to the moved ones).

Examples


UV Sphere
Mesh
Random Value
Value
Extrude Mesh
Mesh
Top
Side
Mesh
Selection
Offset
Set Material
Geometry
Geometry
Selection
Set Material
Geometry
Geometry
Selection
Group Output
Geometry

Segments 16
Rings 8
Radius 1 m
Boolean
Probability 0.400
ID
Seed 0
Faces
Offset 1.000
☑ Individual
Red
Green

Here, the selection outputs are used to set materials on certain faces of the mesh. A Random Value Node is used to limit the extrusion to a random set of faces.

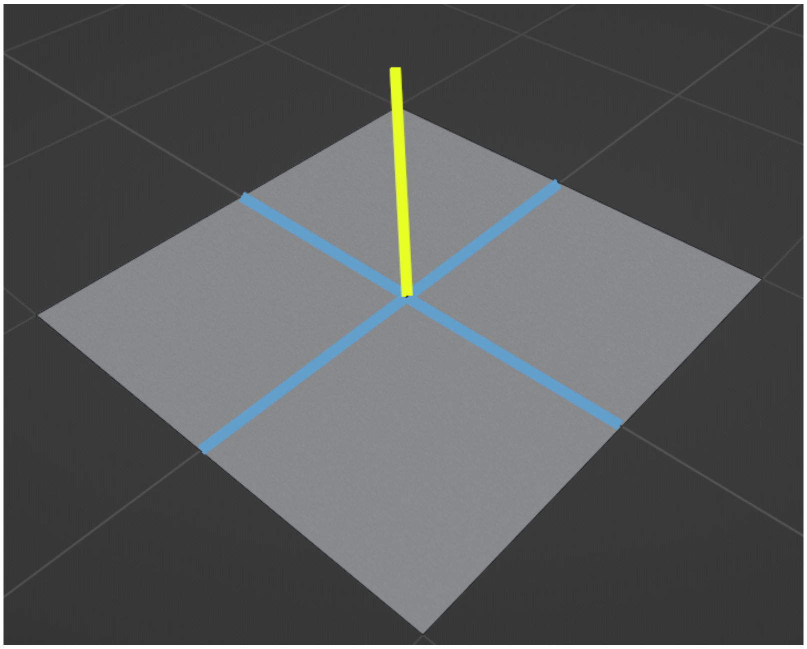
Attribute Propagation

Attributes are transferred to the new elements with specific rules. An attribute will never change domains on the resulting mesh. The `id` attribute does not have any special handling.

Generally boolean attributes are propagated with “or”, meaning any connected “true” value that is mixed in for other types will cause the new value to be “true” as well.

The following sections describe:

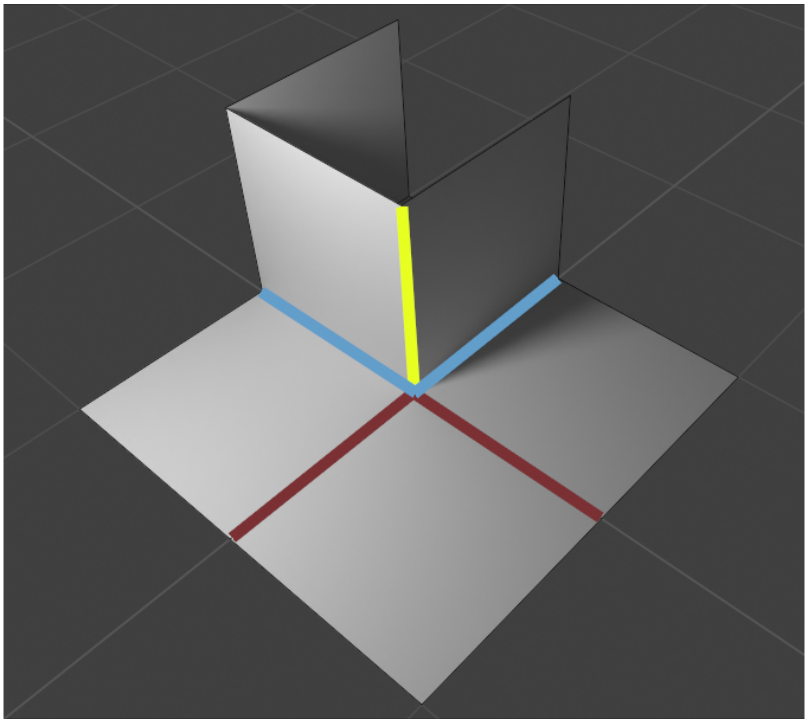
Vertex Mode



The new edges created in vertex mode use the average value of all connected edges.

- New **vertices** have copied values from their original vertices.
- New **edges** have the average value of any connected original edges. For boolean attributes, edges are selected if any connected edges were selected.

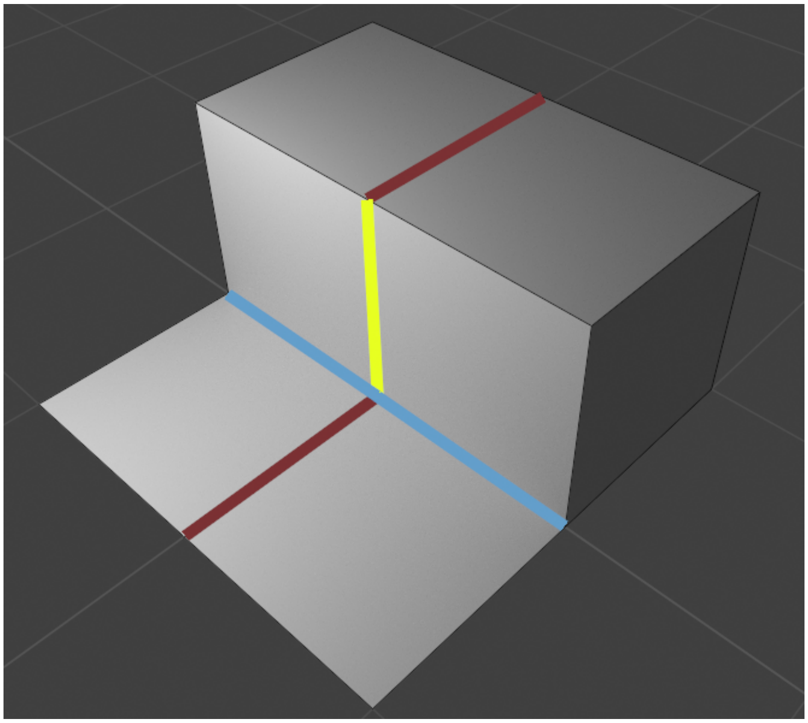
Edge Mode



Attribute propagation for new connecting edges (the vertical yellow edge). The final value is a mix of the values from the two middle blue edges. The darker maroon edges lower on the image are not used.

- New **vertices** have copied values from their original vertices.
- Vertical connecting **edges** get the average value from any connected extruded edges. For booleans, the edges are selected if any connected extruded edges were selected. (*Propagation rules are shown in the figure above*.)
- Horizontal duplicate **edges** have copied values from their original edges.
- New **faces** get the average values of all faces connected to the selected edge. For booleans, faces are selected if any connected original faces were selected.
- New **face corners** get the averaged value of corresponding corners in all faces connected to selected edges. For booleans, corners are selected if one of those corners are selected.

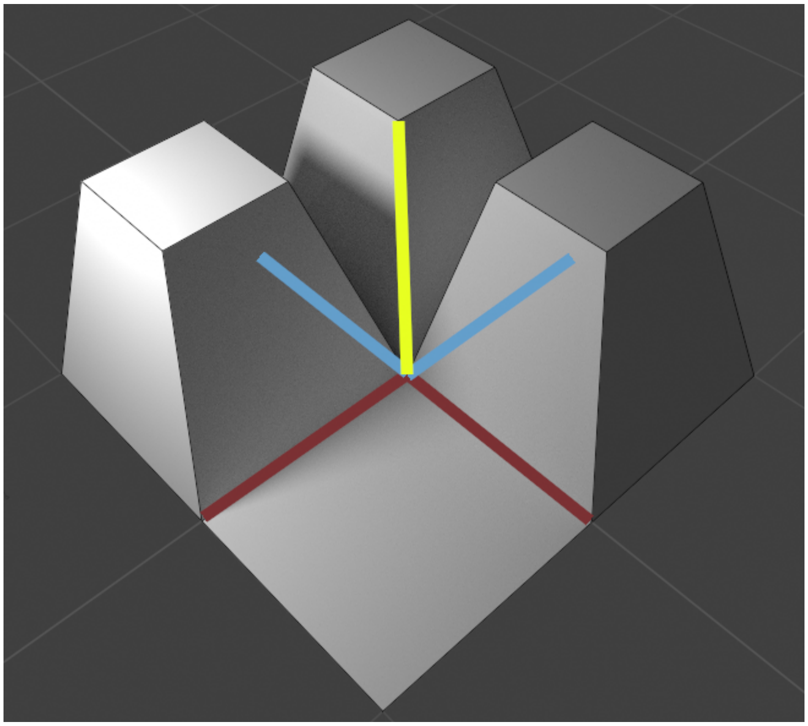
Face Mode



Attribute propagation for new connecting edges (the vertical yellow edge). The final value is a mix of the values from the two middle blue edges. The values from the darker maroon edges between unselected faces and on top of the extruded region are not used.

- New **vertices** have copied values from their original vertices.
- Vertical connecting **edges** get the average value from any connected extruded edges, not including the edges “on top” of extruded regions. For booleans, the edges are selected if any of those connected edges were selected. (*Propagation rules are shown in the figure above*.)
- Horizontal duplicate **edges** have copied values from their original edges.
- New **faces** have copied values from the corresponding extruded faces.
- New **face corners** have copied values from the corresponding corresponding corners of extruded faces.

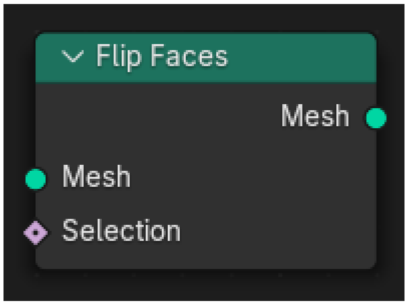
Individual Face Mode



Attribute propagation for new connecting edge. Each edge uses the average values of the two neighboring edges on its extruded face.

- New **vertices** have copied values from their original vertices.
- Vertical connecting **edges** get the average value of the two neighboring edges on each extruded face. For booleans, the edges are selected when at least one neighbor on the extruded face was selected.
- Horizontal duplicate **edges** have copied values from their original edges.
- New side **faces** have copied values from their corresponding selected face.
- New **face corners** have copied values from the corresponding corners of selected faces.

Flip Faces Node



The *Flip Faces Node* reverses the order of the vertices and edges of each selected face. The most common use of this node is to flip the normals of a face. Any face corner domain attributes of selected faces are also reversed.

Though this node is usually used to affect normals, it is not called “Flip Normals” for an important reason. The node does not actually interact with normals directly. Normals are defined by the right hande rule, so if a face’s vertex list is reversed, then its normal will point in the opposite direction.

Inputs

Mesh
Standard geometry input.

Selection
Whether to flip the direction of each face. True values mean the face will be flipped, false means the face will be unaffected.

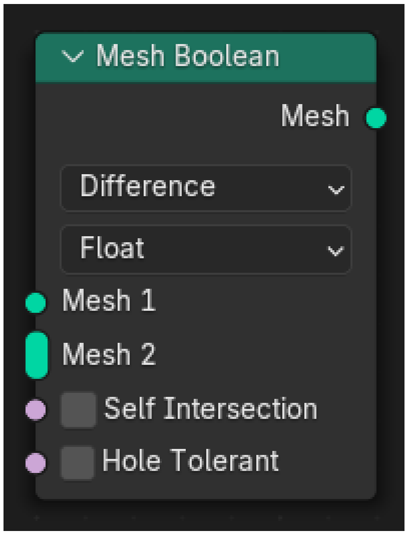
Properties

This node has no properties.

Output

Mesh
Standard geometry output.

Mesh Boolean Node


Mesh Boolean
Mesh
Difference
Float
Mesh 1
Mesh 2
Self Intersection
Hole Tolerant

The *Mesh Boolean Node* allows you to cut, subtract, and join the geometry of two inputs. This node offers the same operations as the Boolean modifier.

Inputs

Mesh 1/2
> Standard geometry input.

Self Intersection
> Correctly calculates cases when one or both operands have self-intersections. This involves more calculations making the node slower.

Hole Tolerant
> Optimizes the Boolean output for Non-manifold geometry at the cost of increased computational time. Because of the performance impact, this option should only be enabled when the solver demonstrates errors with non-manifold geometry.

Properties

Operation

Intersect::	Produce a new geometry containing only the volume inside of both geometry 1 and geometry 2.
Union::	The two input meshes are joined, then any interior elements are removed.
Difference::	Geometry 2 is subtracted from geometry 1 (everything outside of geometry 2 is kept).

Solver

Algorithm used to calculate the Boolean intersections.

Float::	Uses a mathematically simple solver which offers the best performance; however, this solver lacks support for overlapping geometry.
Exact::	Uses a mathematically complex solver which offers the best results and has full support for overlapping geometry; however, this solver is much slower than the *Float Solver*.

Output

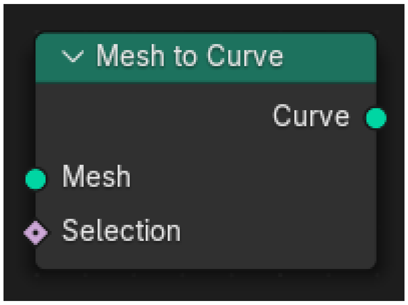
Mesh

Standard geometry output.

Intersecting Edges *Exact Solver*

A boolean attribute field with a selection of the edges that were created where the two inputs meet.

Mesh to Curve Node



The *Mesh to Curve* node turns each string of connected mesh edges into a poly spline. Whenever two or more strings cross each other, the splines will be split.

Loose vertices are ignored – they will not be turned into single-point splines.

Attributes, both named and unnamed ones, are transferred to the resulting splines. If there is a `radius` attribute, it will be applied as such, although you may find it more convenient to use the Set Curve Radius Node for this.

Inputs

Mesh

> Standard mesh input.

Selection

> A field input evaluated on the edge domain to determine whether each edge will be included in the result.
>
> **Tip**
>
> Using this input is more efficient than deleting parts of the geometry before or after the conversion.

Properties

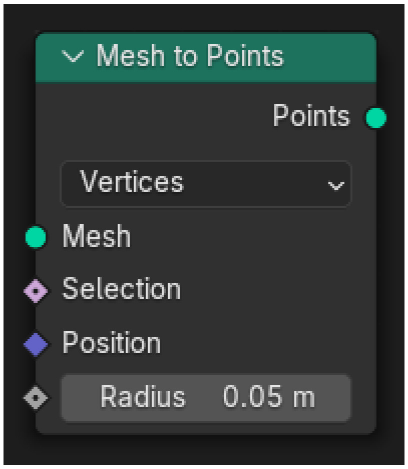
This node has no properties.

Outputs

Curve

Generated curve.

Mesh to Points Node


Mesh to Points
Points
Vertices
Mesh
Selection
Position
Radius 0.05 m

The *Mesh to Points* node generates a point cloud from a mesh.

Inputs

Mesh

Standard Mesh input.

Selection

The meshes used to generate a point cloud.

Position

Positions of generated points. By default, this input is the same as if the [Position Node]() was connected.

Radius

Radii of generated points.

Properties

Mode

Vertices::	Points are generated for each vertex.
Edges::	Points are generated for each edge, at the middle of each edge, by default.

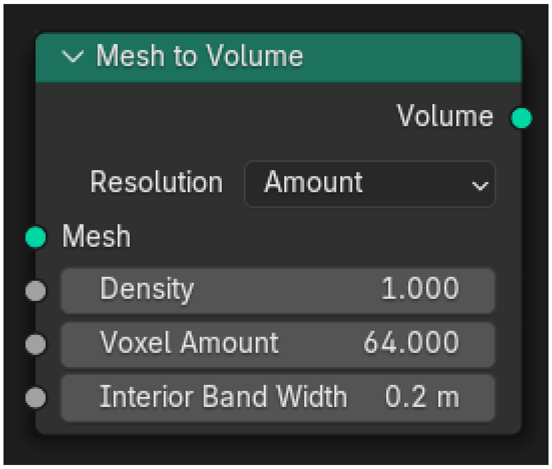
Faces::	Points are generated for each face, at the average of all of each face’s vertices, by default.
Corners::	Points are generated for each corner. The points are all placed at the location of each corners vertex, so they will overlap by default.

Outputs

Points

Generated point cloud.

Mesh to Volume Node


Mesh to Volume
Volume
Resolution
Amount
Mesh
Density
1.000
Voxel Amount
64.000
Interior Band Width
0.2 m

The *Mesh to Volume* node creates a fog volumes based on the shape of a mesh. The volume is created with a grid of the name `"density"`.

Inputs

Mesh
: Standard Mesh input.

Density
: Value of voxels inside the generated fog volume.

Voxel Amount
: Specify the approximate number of voxels along the diagonal.

Voxel Size
: Specify the voxel side length.

Interior Band Width
: The maximum distance of the included voxels to the surface on the inside of the mesh.

Properties

Resolution

How the voxel size is specified.

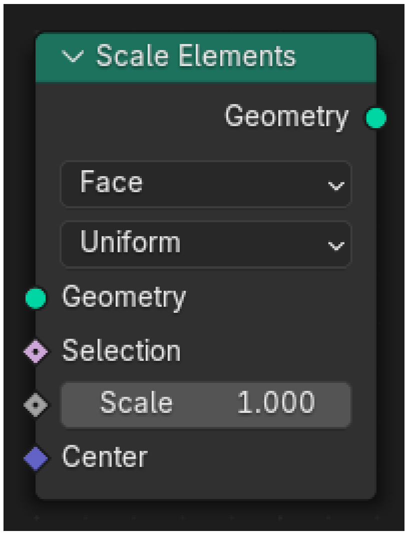
Amount::	Specify the approximate number of voxels along the diagonal.
Size::	Specify the voxel side length. It is recommended to be careful when tweaking this value, because small changes can have a large effect on the processing time.

Outputs

Volume

The generated volume grid.

Scale Elements Node


Scale Elements
Geometry
Face
Uniform
Geometry
Selection
Scale 1.000
Center

Scales the selected faces or edges, letting you specify a scaling factor and pivot point for each one. Connected faces/edges are scaled together using their average factor and pivot point.

Inputs

Geometry
: Standard geometry input.

Selection
: Boolean field indicating which elements to scale.

Scale
: The scaling factor for each element.

Center
: The pivot point for each element.

Axis *Single Axis Mode Only*
: Axis along which to scale each element. This vector is normalized internally, so the length does not matter.

Properties

Domain

The element type to transform.

Face::	Scale faces.
Edge::	Scale edges.

Scale Mode

Uniform::	Scale elements by the same factor in every direction.
Single Axis::	Scale elements in a single direction defined by the *Axis* input.

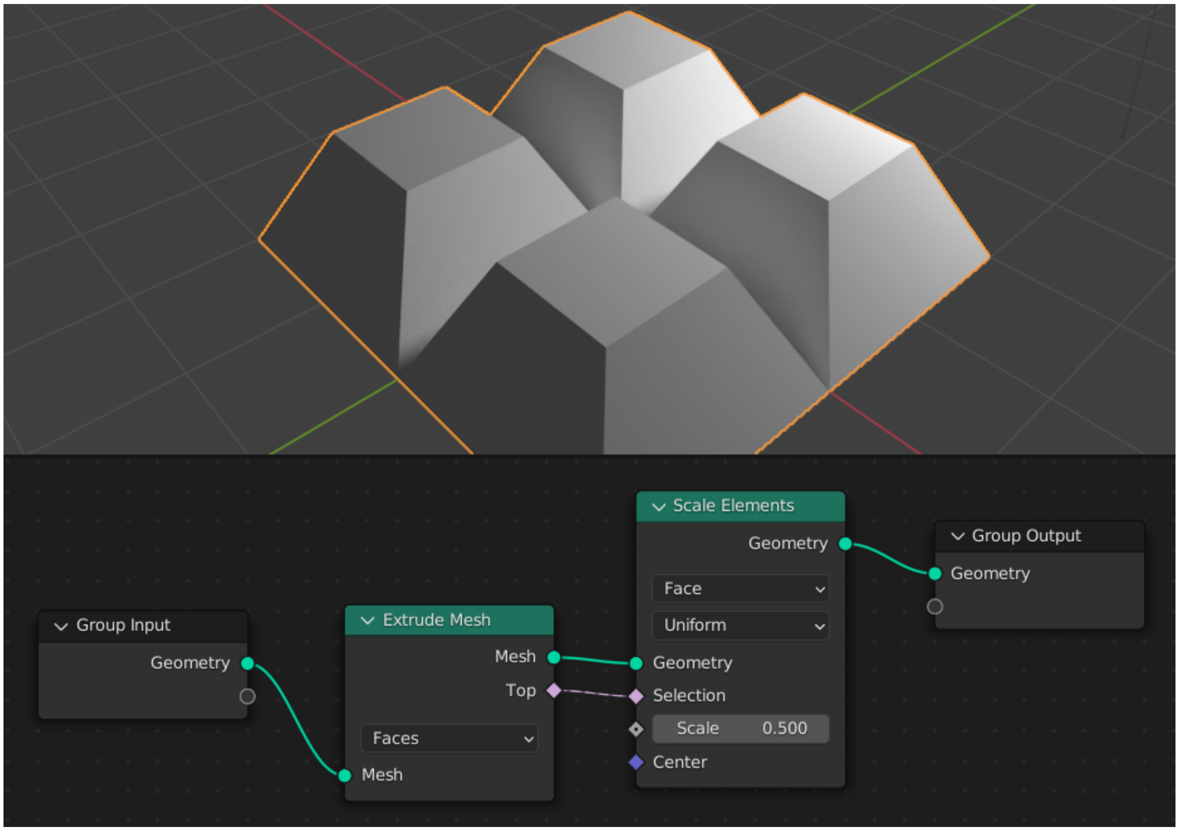
Output

Geometry

Standard geometry output.

Examples

The node is useful when combined with the Extrude Mesh Node, especially in *Individual* mode where connected faces aren’t extruded together.

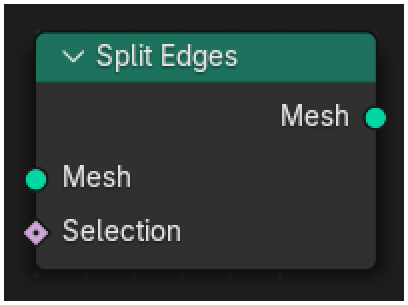

Group Input
Geometry

Extrude Mesh
Mesh
Top
Faces
Mesh

Scale Elements
Geometry
Face
Uniform
Geometry
Selection
Scale 0.500
Center

Group Output
Geometry

Split Edges Node



Like the Edge Split Modifier, the *Split Edges* node splits and duplicates edges within a mesh, breaking ‘links’ between faces around those split edges.

Inputs

Mesh
: Standard geometry input.

Selection
: A standard Boolean selection input to determine which edges will be split.

Note

Because of mesh topology requirements, sometimes more or fewer edges than are selected will be split.

Properties

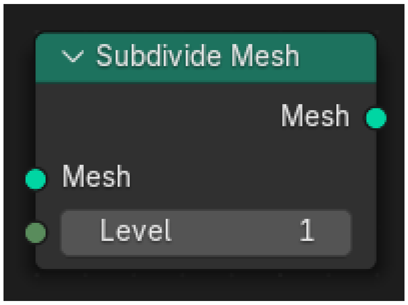
This node has no properties.

Outputs

Mesh
: Standard geometry output.

Subdivide Mesh Node

The *Subdivide Mesh* node adds new faces to mesh geometry using a simple interpolation for deformation.



Inputs

Mesh

Standard geometry input.

Level

The number of subdivisions to apply to the input geometry.

Properties

This node has no properties.

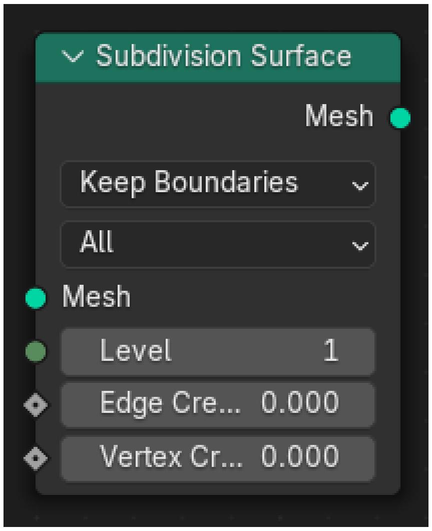
Outputs

Mesh

Standard geometry output.

Subdivision Surface Node

The *Subdivision Surface* node adds new faces to mesh geometry using a Catmull-Clark subdivision method.



Keep Boundaries

All

Level 1

Edge Cre... 0.000

Vertex Cr... 0.000

Inputs

Mesh

Standard geometry input.

Level

The number of subdivisions to apply to the input geometry.

Edge Crease

Controls how smooth edges should be with Weighted Edge Creases.

Vertex Crease

Controls how much the subdivision surface should be pulled towards the vertex. Similar to edge creases, but allows individual vertices to be creased.

Properties

UV Smooth

Controls how subdivision smoothing is applied to UVs.

None::	UVs remain unchanged.
Keep Corners::	UV islands are smoothed, but their boundary remain unchanged.
Keep Corners, Junctions::	UVs are smoothed, corners on discontinuous boundary and junctions of three or more regions are kept sharp.
Keep Corners, Junctions, Concave::	UVs are smoothed, corners on discontinuous boundary, junctions of three or more regions and darts and concave corners are kept sharp.
Keep Boundaries::	UVs are smoothed, boundaries are kept sharp.
All::	UVs and their boundaries are smoothed.

Boundary Smooth

Controls how open boundaries (and corners) are smoothed.

All::	Smooth boundaries, including corners.
Keep Corners::	Smooth boundaries, but corners are kept sharp.

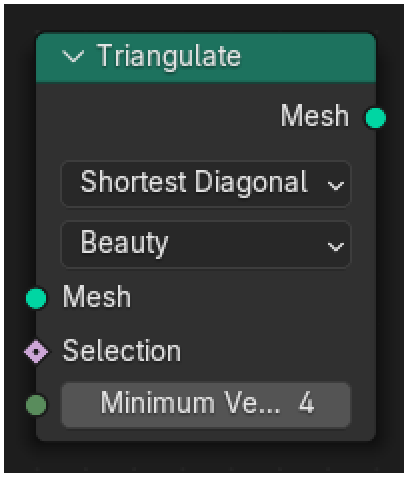
Outputs

Mesh

Standard geometry output.

Triangulate Node

The *Triangulate* node converts all faces in a mesh (quads and n-gons) to triangular faces. It functions the same as the Triangulate tool in Edit Mode.



Inputs

Mesh
: Standard geometry input.

Selection
: A standard Boolean selection input to determine which faces will be triangulated.

Minimum Vertices
: Minimum number of vertices a face must have to be triangulated. For example, setting this value to 5, will prevent triangulation of Quads and only triangulate N-gons.

Properties

Quad Method

Beauty::	Split the quads in nice triangles, slower method.
Fixed::	Split the quads on their 1st and 3rd vertices.
Fixed Alternate::	Split the quads on their 2nd and 4th vertices.

Shortest Diagonal::	Split the quads along their shortest diagonal.
Longest Diagonal::	Split the quads along their longest diagonal. This is the preferred mode for cloth simulations.

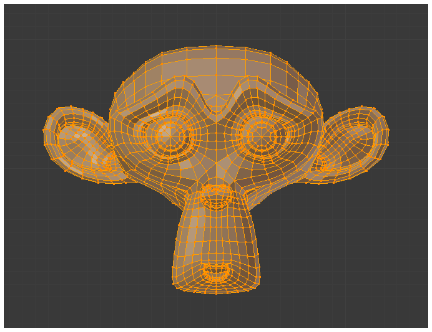
N-gon Method

Beauty::	Arrange the new triangles nicely, slower method.
Clip::	Split n-gons using an ear-clipping algorithm (the same method of tessellation used for viewport display).

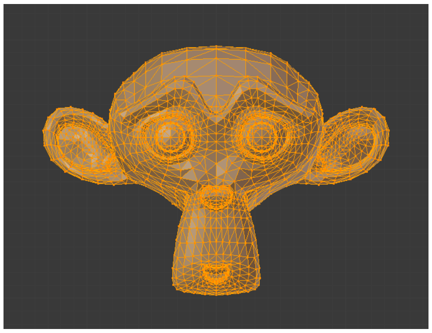
Outputs

Mesh
  Standard geometry output.

Example



Mesh before triangulation.



Mesh after triangulation.

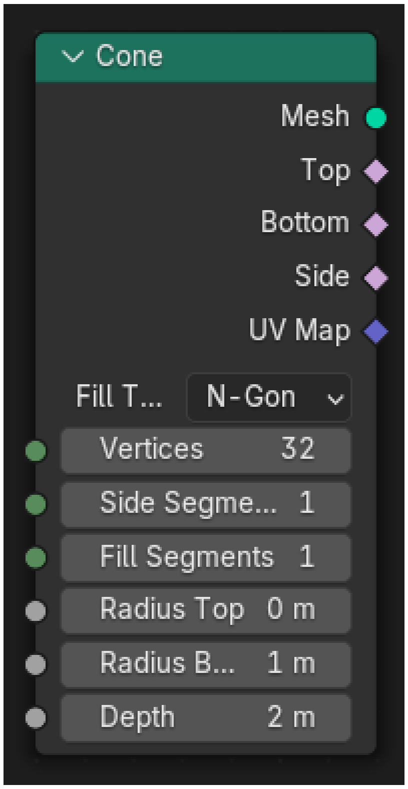
Mesh Primitive Nodes

Nodes that create a primitive mesh, e.g. a cube.

- Cone Node
- Cube Node
- Cylinder Node
- Grid Node
- Icosphere Node
- Mesh Circle Node
- Mesh Line Node
- UV Sphere Node

Cone Node

Generates a cone mesh that is optionally truncated.



Fill T... N-Gon

Vertices 32

Side Segme... 1

Fill Segments 1

Radius Top 0 m

Radius B... 1 m

Depth 2 m

Inputs

Vertices
: Number of vertices in the top and/or bottom circle of the cone. No geometry is generated if the number is below three.

Side Segments
: Number of vertically stacked face loops that make up the cone’s sides. Increasing this will add horizontal cuts. No geometry is generated if the number is below one.

Fill Segments
: Number of concentric rings in the top and/or bottom. No geometry is generated if the number is below one.

Radius Top

The radius of the cone's top circle. If this is zero, the circle is reduced to a single vertex.

Radius Bottom

Same as *Radius Top* but for the bottom circle.

Depth

Height of the generated cone.

Note

If the top and bottom radii are both zero, this node will output a single line.

Properties

Fill Type

How the circles at the top and bottom are filled with faces when their radius is larger than zero.

None::	Do not fill the circles.
N-Gon::	Fill the innermost circles with a single face.
Triangles::	Fill the innermost circles with triangles connected to a vertex in the center.

Outputs

Mesh

Standard geometry output.

Top

A boolean field with a selection of the faces on the top of the cone. If the *Fill Type* is set to *None*, this will be a selection of the top edges instead. If *Radius Top* is zero, this will be a selection of the top vertex.

Side

A boolean field with a selection of the faces on the side of the cone.

Bottom

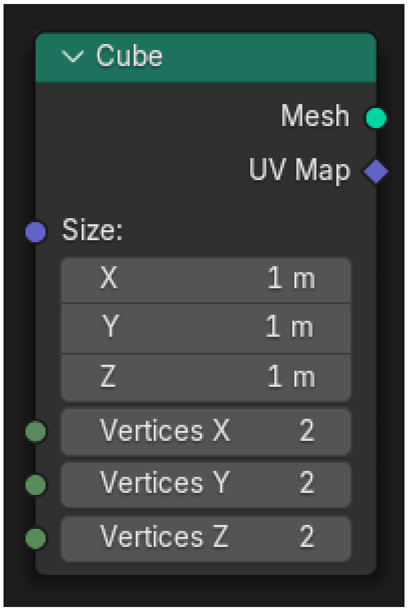
A boolean field with a selection of the faces on the bottom of the cone. If the *Fill Type* is set to *None*, this will be a selection of the bottom edges instead. If *Radius Bottom* is zero, this will be a selection of the bottom vertex.

UV Map

The default UV coordinate of each face corner. This can be connected to the Store Named Attribute Node for populating a UV Map.

Cube Node

The *Cube* node generates a cuboid mesh with variable side lengths and subdivisions. The inside of the mesh is still hollow like a normal cube.


Cube
Mesh
UV Map
Size:
X 1 m
Y 1 m
Z 1 m
Vertices X 2
Vertices Y 2
Vertices Z 2

Inputs

Size
: Side lengths along each of the main axes.

Vertices X, Y, Z
: Number of vertices for each side of the cube. The number of vertices should be at least 1.

Properties

This node has no properties.

Outputs

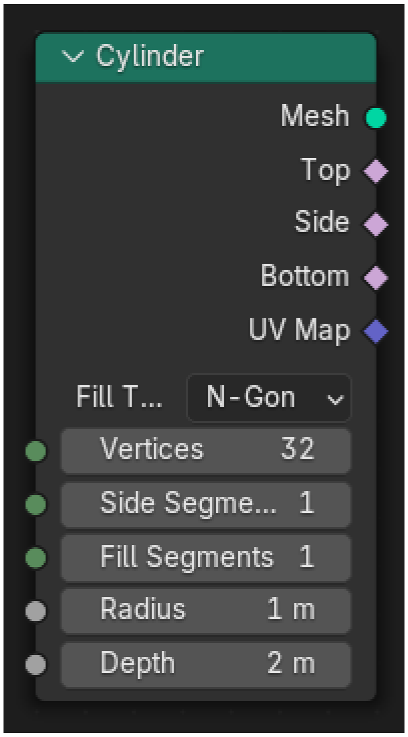
Mesh
: Standard geometry output.

UV Map

A 2D vector representing the default X/Y coordinates of the UV Map for the primitive’s shape. This can be connected to the Store Named Attribute Node, to be used once the Geometry Nodes Modifier get applied. The UV map must be stored on the face corner in order to be accessed.

Cylinder Node

The *Cylinder* node generates a cylinder mesh. It is similar to the Cone node but always uses the same radius for the circles at the top and bottom.


Cylinder
Mesh
Top
Side
Bottom
UV Map
Fill T...
N-Gon
Vertices 32
Side Segme... 1
Fill Segments 1
Radius 1 m
Depth 2 m

Inputs

Vertices
: Number of vertices on the circle at the top and bottom. No geometry is generated if the number is below three.

Side Segments
: Number of edges running vertically along the side of the cone. No geometry is generated if the number is below one.

Fill Segments
: Number of concentric rings used to fill the round faces at the top and bottom. No geometry is generated if the number is below one.

Radius

Distance of the vertices from the Z axis. If this is zero, the output will be a single line.

Depth

Height of the cylinder.

Properties

Fill Type

How the circles at the top and bottom are filled with faces when their radius is larger than zero.

None::	Do not fill the circles.
N-Gon::	Fill the innermost segment of the circles with a single face.
Triangles::	Fill the innermost segment of the circles with triangles connected to a new vertex on the Z axis.

Outputs

Mesh

Standard geometry output.

Top

A boolean attribute field with a selection of the faces on the top of the cylinder. If the *Fill Type* property is *None*, then this will be a selection of the top edges instead. If the *Radius* is zero, this will be a selection of the top point.

Side

A boolean attribute field with a selection of the faces on the side of the cylinder.

Bottom

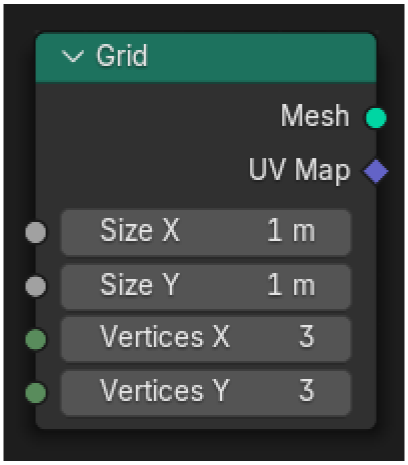
This is the same as the *Top* selection output, but on the bottom side of the geometry instead.

UV Map

A 2D vector representing the default X/Y coordinates of the UV Map for the primitive’s shape. This can be connected to the Store Named Attribute Node, to be used once the Geometry Nodes Modifier get applied. The UV map must be stored on the face corner in order to be accessed.

Grid Node

The *Grid* node generates a planar mesh on the XY plane.



Size X 1 m
Size Y 1 m
Vertices X 3
Vertices Y 3

Inputs

Size X
: Side length of the plane in the X direction.

Size Y
: Side length of the plane in the Y direction.

Vertices X
: Number of vertices in the X direction. If this is smaller than two, no mesh is generated.

Vertices Y
: Number of vertices in the Y direction. If this is smaller than two, no mesh is generated.

Properties

This node has no properties.

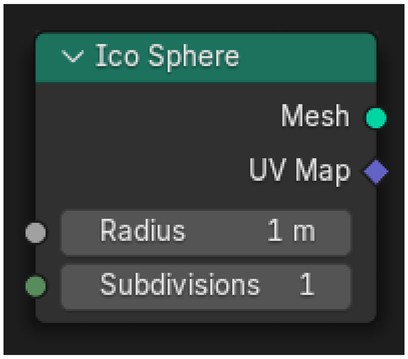
Outputs

Mesh
: Standard geometry output.

UV Map
: A 2D vector representing the default X/Y coordinates of the UV Map for the primitive's shape. This can be connected to the Store Named Attribute Node, to be used once the Geometry Nodes Modifier get applied. The UV map must be stored on the face corner in order to be accessed.

Icosphere Node

The *Icosphere* node generates a spherical mesh that consists of equally sized triangles.



Radius 1 m

Subdivisions 1

Inputs

Radius

Distance of the vertices from the origin.

Subdivisions

Number of subdivisions on top of the most basic icosphere. The number of faces quadruple with every subdivision.

Properties

This node has no properties.

Outputs

Mesh

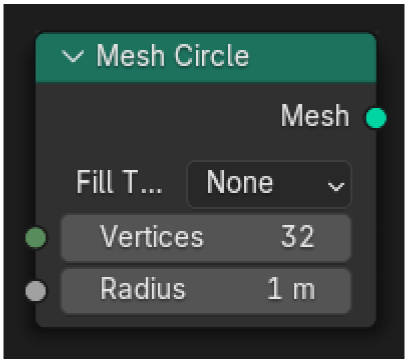
Standard geometry output.

UV Map

A 2D vector representing the default X/Y coordinates of the UV Map for the primitive’s shape. This can be connected to the Store Named Attribute Node, to be used once the Geometry Nodes Modifier get applied. The UV map must be stored on the face corner in order to be accessed.

Mesh Circle Node

The *Mesh Circle* node generates a circular ring of edges that is optionally filled with faces.



Fill T... None

Vertices 32

Radius 1 m

Inputs

Vertices

Number of vertices on the circle. No geometry is generated when the number is below three.

Radius

Distance of the vertices from the origin.

Properties

Fill Type

How the circle is filled with faces.

None::	Output just the edge ring without any faces.
N-Gon::	Fill the circle with a single face.
Triangles::	Fill the circle with triangles connected to a new vertex at the origin.

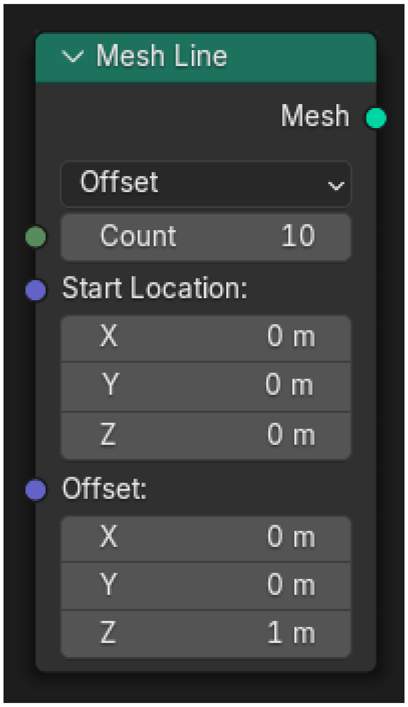
Outputs

Mesh

Standard geometry output.

Mesh Line Node

The *Mesh Line* node generates vertices in a line and connects them with edges.



Offset
Count 10
Start Location:
X 0 m
Y 0 m
Z 0 m
Offset:
X 0 m
Y 0 m
Z 1 m

Inputs

Count
: Number of vertices on the line.

Resolution
: Length of individual edges. The node tries to fit as many vertices as possible between the start and end point. The exact end point might not be hit. This is only available when the mode is set to *End Points* and the count mode is set to *Resolution*.

Start Location
: Position of the first vertex.

Offset

Controls the direction of the line and distance between the vertices. This is only available when the mode is set to *Offset*.

End Location

Position of the last vertex. This is only available when the mode is set to *End Points*.

Properties

Mode

Inputs to use to control the line.

Offset::	Specify the offset from one vertex to the next.
End Points::	Specify the start and end point of the line.

Count Mode

Determines how the number of vertices is chosen. This is only available when the mode is set to *End Points*.

Count::	Specify the total number of vertices.
Resolution::	Specify the distance between vertices.

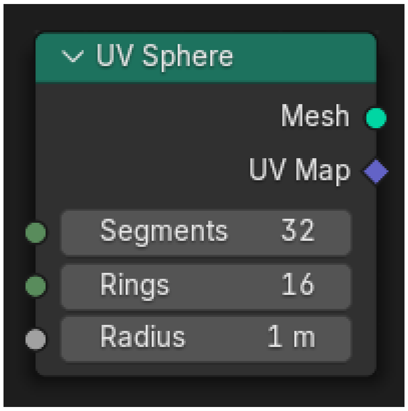
Outputs

Mesh

Standard geometry output.

UV Sphere Node

The *UV Sphere* node generates a spherical mesh mostly out of quads except for triangles at the top and bottom.



Segments 32
Rings 16
Radius 1 m

Inputs

Segments
: Horizontal resolution of the sphere. If this is smaller than three, no mesh is generated.

Rings
: Vertical resolution of the sphere. If this is smaller than two, no mesh is generated.

Radius
: Distance of vertices to the origin.

Properties

This node has no properties.

Outputs

Mesh
: Standard geometry output.

UV Map

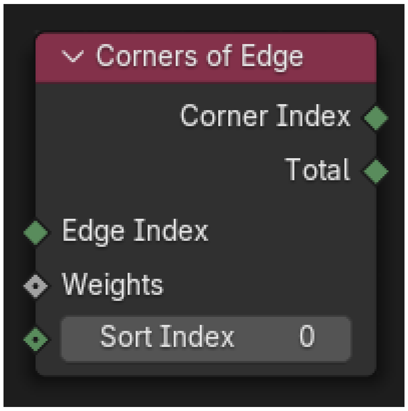
A 2D vector representing the default X/Y coordinates of the UV Map for the primitive's shape. This can be connected to the Store Named Attribute Node, to be used once the Geometry Nodes Modifier get applied. The UV map must be stored on the face corner in order to be accessed.

Mesh Topology Nodes

Nodes that retrieve information about the connectivity between mesh elements.

- Corners of Edge Node
- Corners of Face Node
- Corners of Vertex Node
- Edges of Corner Node
- Edges of Vertex Node
- Face of Corner Node
- Offset Corner in Face Node
- Vertex of Corner Node

Corners of Edge Node



Selects a neighboring face corner of an edge and outputs its index.

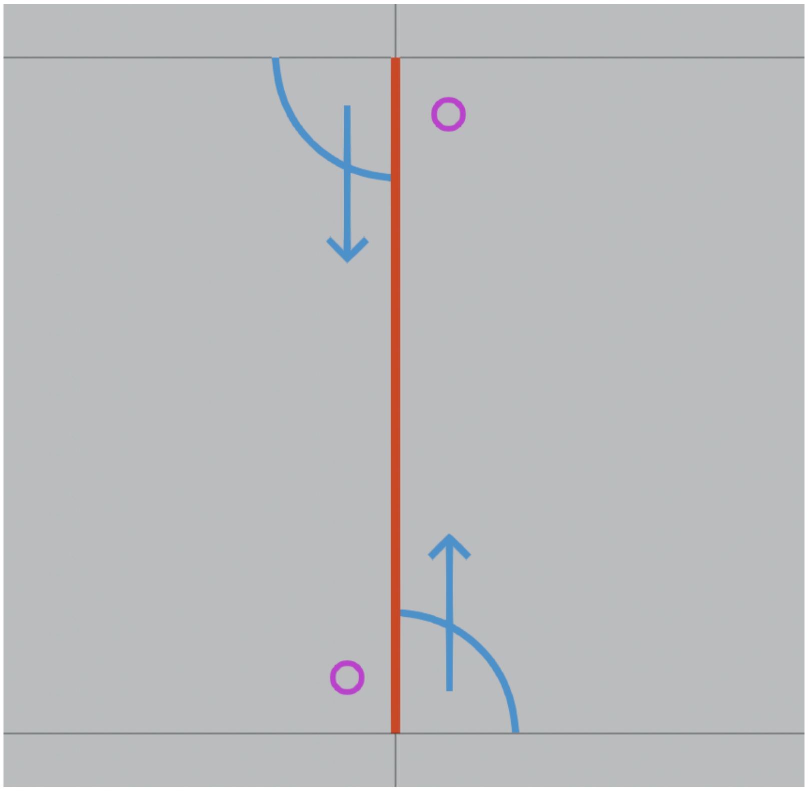
This node is a bit special because it operates in two different domains. First, it evaluates a *Weight* for each corner in the geometry. Then, for each item in the context domain, it will:

- Pick an edge from the geometry based on the *Edge Index*.
- Find *some* (not all) face corners connected to this edge – see below.
- Sort these corners by their associated weight.
- Pick a corner from the above sorted list based on the *Sort Index*, where 0 means the corner with the lowest weight, 1 means the corner with the second-lowest weight and so on.
- Output the geometry-wide index of this corner.

Warning

As illustrated below, the node only looks at one corner per connected face. Even though the edge has four neighboring corners, *Corner Index* can only return the indexes of two of them, and *Total* will similarly return 2.

You can use the Offset Corner in Face Node to retrieve the indexes of the other corners.



A graphic for which corners are returned for a given edge

- Red: selected edge.
- Blue: the corners whose index can be retrieved using this node.
- Purple: the corners that can be retrieved by offsetting the blue corner indices using the Offset Corner in Face Node.

Inputs

Edge Index

The index of the edge for which to find connected corners.

Note

If this input is not connected, it uses the index of the context item, which means it's important that the node is evaluated in the Edge domain.

Weights

The weights of the corners in the geometry. Unlike the other inputs which follow the context domain, this one is always evaluated in the Face Corner domain.

The corners are sorted by their associated weight in ascending order. Corners with the same weight are sorted by their index.

Sort Index

The 0-based index of the corner to select from the edge's sorted corners. If this value is outside the range of valid indices, it wraps around.

Properties

This node has no properties.

Outputs

Corner Index

The geometry-wide index of the selected corner. You can pass this to the Evaluate at Index Node or the Sample Index Node (with the domain set to Face Corner) to retrieve details about the corner.

If the edge has no connected corners, *Corner Index* will be zero.

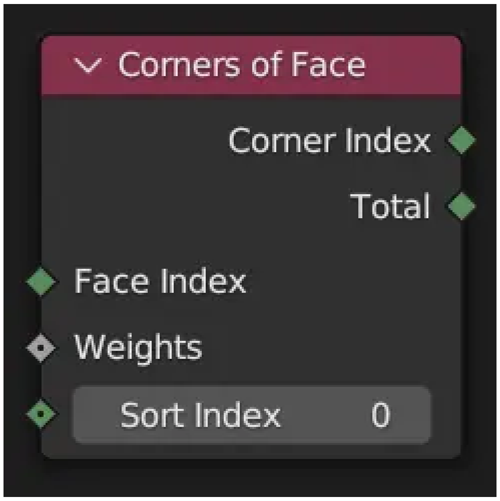
Total

The number of faces (not face corners!) connected to the edge.

See also

The page for the Edges of Vertex Node has an example of how to work with the different domains.

Corners of Face Node



Selects a corner of a face and outputs its index.

This node is a bit special because it operates in two different domains. First, it evaluates a *Weight* for each corner in the geometry. Then, for each item in the context domain, it will:

- Pick a face from the geometry based on the *Face Index*.
- Find the corners of this face.
- Sort these corners by their associated weight.
- Pick a corner from the above sorted list based on the *Sort Index*, where 0 means the corner with the lowest weight, 1 means the corner with the second-lowest weight and so on.
- Output the geometry-wide index of this corner.

Inputs

Face Index

The index of the face for which to find the corners.

Note

If this input is not connected, it uses the index of the context item, which means it's important that the node is evaluated in the Face domain.

Weights

The weights of the corners in the geometry. Unlike the other inputs which follow the context domain, this one is always evaluated in the Face Corner domain.

The corners are sorted by their associated weight in ascending order. Corners with the same weight are sorted by their index.

Sort Index

The 0-based index of the corner to select from the face's sorted corners. If this value is outside the range of valid indices, it wraps around.

Properties

This node has no properties.

Outputs

Corner Index

The geometry-wide index of the selected corner. You can pass this to the Evaluate at Index Node or the Sample Index Node (with the domain set to Face Corner) to retrieve details about the corner.

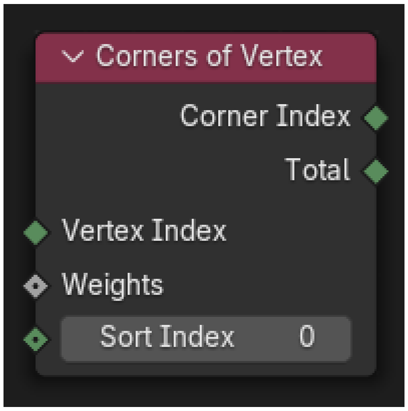
Total

The number of corners in the face, which is also its number of edges.

See also

The page for the Edges of Vertex Node has an example of how to work with the different domains.

Corners of Vertex Node



Selects a neighboring face corner of a vertex and outputs its index.

This node is a bit special because it operates in two different domains. First, it evaluates a *Weight* for each corner in the geometry. Then, for each item in the context domain, it will:

- Pick a vertex from the geometry based on the *Vertex Index*.
- Find the face corners adjacent to this vertex.
- Sort these corners by their associated weight.
- Pick a corner from the above sorted list based on the *Sort Index*, where 0 means the corner with the lowest weight, 1 means the corner with the second-lowest weight and so on.
- Output the geometry-wide index of this corner.

Inputs

Vertex Index

The index of the vertex for which to find the corners.

> **Note**
>
> If this input is not connected, it uses the index of the context item, which means it's important that the node is evaluated in the Point domain.

Weights

The weights of the corners in the geometry. Unlike the other inputs which follow the context domain, this one is always evaluated in the

Face Corner domain.

The corners are sorted by their associated weight in ascending order. Corners with the same weight are sorted by their index.

Sort Index

The 0-based index of the corner to select from the vertex's sorted corners. If this value is outside the range of valid indices, it wraps around.

Properties

This node has no properties.

Outputs

Corner Index

The geometry-wide index of the selected corner. You can pass this to the Evaluate at Index Node or the Sample Index Node (with the domain set to Face Corner) to retrieve details about the corner.

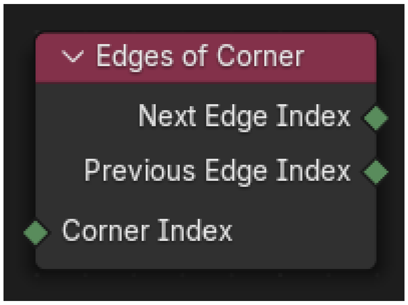
Total

The number of adjacent corners, which is also the number of faces.

See also

The page for the Edges of Vertex Node has an example of how to work with the different domains.

Edges of Corner Node


Edges of Corner
Next Edge Index
Previous Edge Index
Corner Index

The *Edges of Corner* node retrieves the edges on both sides of a face corner.

Inputs

Corner Index
: The index of the input face corner.

> **Note**
>
> By default this uses the index from the field context, which makes it important that the node is evaluated on the face corner domain.

Properties

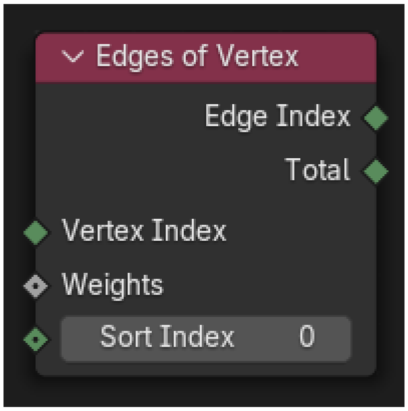
This node has no properties.

Outputs

Next Edge Index
: The index of the neighboring edge in the face, in the direction of increasing face corner indices.

Previous Edge Index
: The index of the neighboring edge in the face, in the direction of decreasing face corner indices.

Edges of Vertex Node



Selects a neighboring edge of a vertex and outputs its index.

This node is a bit special because it operates in two different domains. First, it evaluates a *Weight* for each edge in the geometry. Then, for each item in the context domain, it will:

- Pick a vertex from the geometry based on the *Vertex Index*.
- Find the edges connected to this vertex.
- Sort these edges by their associated weight.
- Pick an edge from the above sorted list based on the *Sort Index*, where 0 means the edge with the lowest weight, 1 means the edge with the second-lowest weight and so on.
- Output the geometry-wide index of this edge.

Inputs

Vertex Index

The index of the vertex for which to find the edges.

Note

If this input is not connected, it uses the index of the context item, which means it's important that the node is evaluated in the Point domain.

Weights

The weights of the edges in the geometry. Unlike the other inputs which follow the context domain, this one is always evaluated in the Edge domain.

The edges are sorted by their associated weight in ascending order. Edges with the same weight are sorted by their index.

Sort Index

The 0-based index of the edge to select from the vertex's sorted edges. If this value is outside the range of valid indices, it wraps around.

Properties

This node has no properties.

Outputs

Edge Index

The geometry-wide index of the selected edge. You can pass this to the Evaluate at Index Node or the Sample Index Node (with the domain set to Edge) to retrieve details about the edge.

If the vertex has no connected edges, *Edge Index* will be zero.

Total

The number of edges connected to the selected vertex.

Example

The example below creates a cone at each vertex of a "cube," aligned to the neighboring edge that's the most vertical.

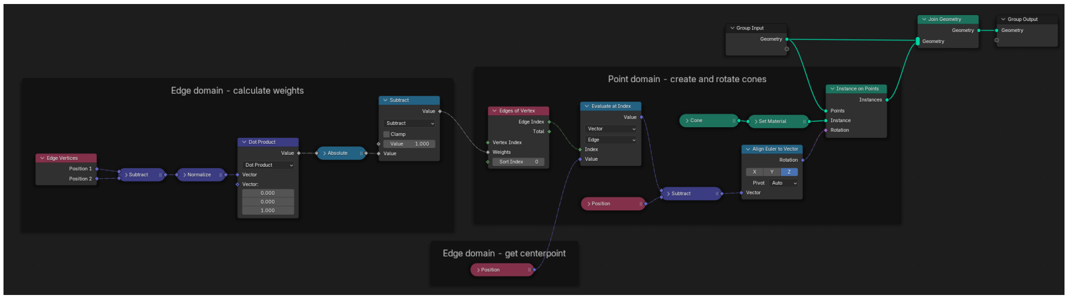
First, we calculate a "verticality score" for each of the cube's edges. To do this, we subtract the positions of its vertices to get its direction vector, which we normalize and use to calculate the dot product with the Z axis. The absolute value of that gives us a number between 0 and 1, where 0 means fully horizontal and 1 means fully vertical.

Because the edges will be sorted by ascending weight, we set weight = 1 - verticality. This way, the most vertical connected edge of each vertex will have the lowest weight and come first in the list.

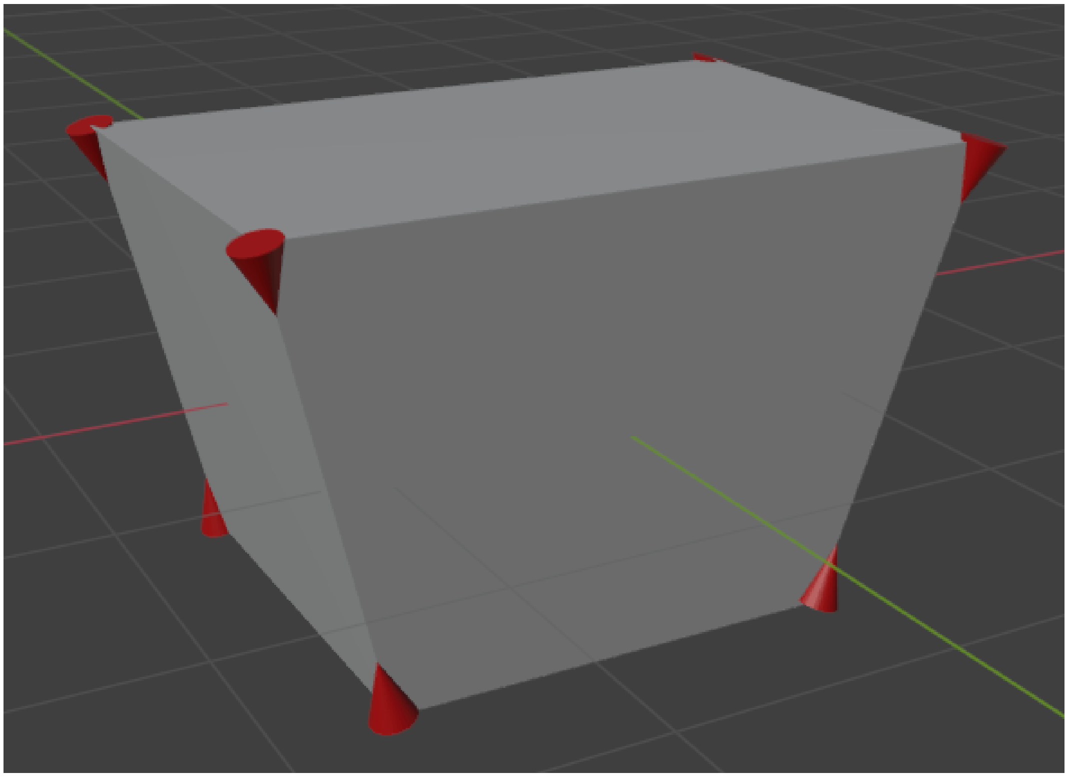
Next, in the point domain, we need to calculate the rotation of each cone. By using the Align Rotation to Vector Node, the problem gets simplified and we only need to calculate a direction vector.

The direction vector of each cone is the centerpoint of the most vertical neighboring edge minus the position of the vertex. Finding that most vertical neighboring edge is where the *Edges of Vertex* node comes in: for each vertex, it sorts the connected edges by their weight and pick the first one (because the Sort Index is 0). Once we have the edge's index, we use the Evaluate at Index Node to retrieve its centerpoint.

With the rotations of the cones calculated, we use the Instance on Points Node to create them.


Edge domain - calculate weights
Edge Vertices
Position 1
Position 2
Subtract
Normalize
Dot Product
Value
Dot Product
Vector
Vector:
0.000
0.000
1.000
Absolute
Subtract
Value
Subtract
Clamp
Value 1.000
Value
Group Input
Geometry
Point domain - create and rotate cones
Edges of Vertex
Edge Index
Total
Vertex Index
Weights
Sort Index 0
Evaluate at Index
Value
Vector
Edge
Index
Value
Cone
Set Material
Position
Subtract
Align Euler to Vector
Rotation
X
Y
Z
Pivot
Auto
Vector
Instance on Points
Instances
Points
Instance
Rotation
Join Geometry
Geometry
Geometry
Group Output
Geometry
Edge domain - get centerpoint
Position

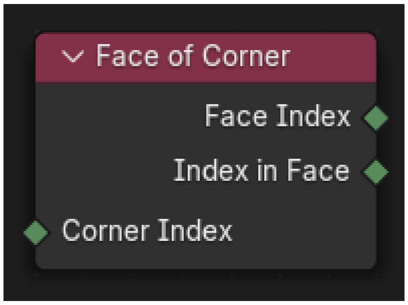
Example node setup. (Rightclick and choose "Open image in new tab" to see a larger version.)



The resulting geometry.

Face of Corner Node

Retrieves the face that a face corner is part of.



Inputs

Corner Index

The geometry-wide index of the corner.

> **Note**
>
> If this input is not connected, it uses the index of the context item, which means it's important that the node is evaluated in the Face Corner domain.

Properties

This node has no properties.

Outputs

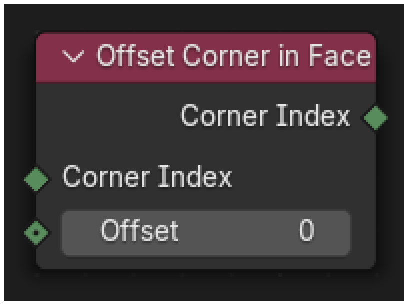
Face Index

The geometry-wide index of the face which the corner belongs to.

Index in Face

The face-local index of the corner. This is 0 for the first corner of the face, 1 for the next corner, and so on up to (number of corners - 1) for the last corner.

Offset Corner in Face Node



Retrieves another corner in the same face as the input corner. This is like “rotating” the input corner around in its face.

Conceptually the operation is similar to the [Offset Point in Curve Node](#).

Inputs

Corner Index

> The index of the input face corner.
>
> **Note**
>
> If this input is not connected, it uses the [index](#) of the context item, which means it’s important that the node is evaluated in the Face Corner domain.

Offset

> The number of corners to move around the face before finding the result, circling back to the first corner if necessary.

Properties

This node has no properties.

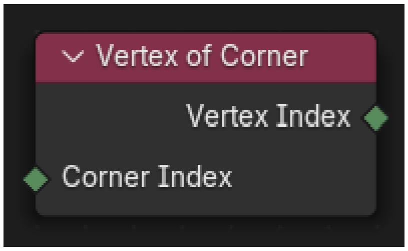
Outputs

Corner Index

> The index of the offset face corner.

Vertex of Corner Node

Outputs the index of the vertex that a face corner is attached to.



Inputs

Corner Index

The index of the face corner.

Note

If this input is not connected, it uses the index of the context item, which means it's important that the node is evaluated in the Face Corner domain.

Properties

This node has no properties.

Outputs

Vertex Index

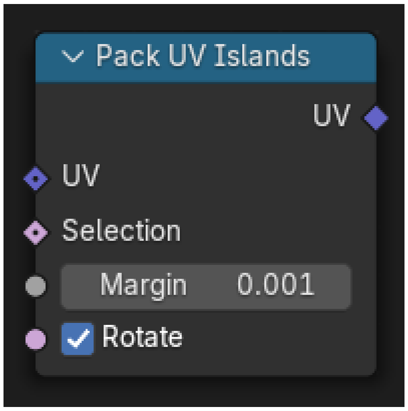
The index of the vertex that the face corner is attached to.

UV Nodes

Nodes for creating or modifying UV Maps.

- Pack UV Islands Node
- UV Unwrap Node

Pack UV Islands Node



The *Pack UV Islands Node* scales islands of a UV map and moves them so they fill the UV space as much as possible.

See also

The Pack Islands operator performs a similar operation in the UV editor.

Inputs

UV
: The UV map to modify.

Selection
: Faces to consider when packing islands. UVs that are part of any other face will not be affected.

Margin
: The distance to leave between UV islands.

Rotate
: Allow Rotating islands for best fit.

Properties

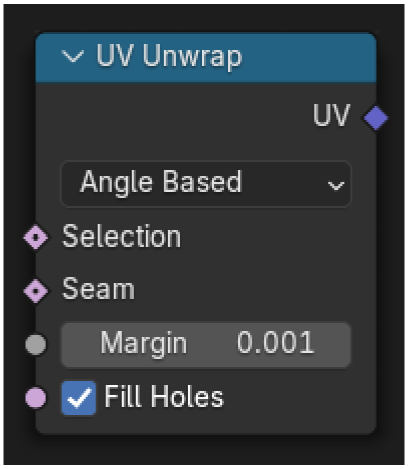
This node has no properties.

Output

UV

The modified UVs.

UV Unwrap Node



The *UV Unwrap Node* generates a UV map islands based on a selection of seam edges. The node implicitly performs a Pack Islands operation upon completion, because the results may not be generally useful otherwise.

See also

The Unwrap operator performs a similar operation in the UV editor. Unlike the Unwrap operator, the node doesn't perform aspect ratio correction, because it is trivial to implement with a Vector Math Node.

Inputs

Selection
: Faces to participate in the unwrap operation. UVs that are part of any other face will not be affected.

Seam
: Edges to mark where the mesh is "cut" for the purposes of unwrapping.

Margin
: The distance to leave between UV islands.

Fill Holes
: Virtually fill holes in mesh before unwrapping, to better avoid overlaps and preserve symmetry.

Properties

Method

Angle Based::	This method gives a good 2D representation of a mesh
Conformal::	Uses LSCM (Least Squares Conformal Mapping). This usually gives a less accurate UV mapping than Angle Based, but works better for simpler objects

Output

UV

The generated UV coordinates between 0 and 1 for each face corner in the selected faces.

Note

In order for Blender to recognize the created attribute as a UV map, it must be created with the Store Named Attribute Node on the Face Corner domain with the 2D Vector data type. This is necessary because there is no 2D Vector socket type.

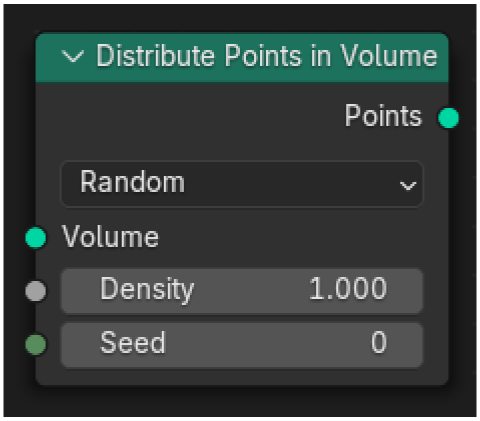
Point Nodes

Nodes generate or modify point clouds.

- Distribute Points in Volume
- Distribute Points on Faces
- Points Node
- Points to Curves Node
- Points to Vertices Node
- Points to Volume Node

- Set Point Radius Node

Distribute Points in Volume


Distribute Points in Volume
Points
Random
Volume
Density 1.000
Seed 0

The *Distribute Points in Volume* node creates points inside of volume grids. The node has two basic modes of operation: distributing points randomly, or in a regular grid. Both methods operate on all of the float grids in the volume.

Inputs

Volume
: Standard volume geometry input.

Density
: Number of points to sample per unit volume.

Spacing
: Spacing between grid points.

Threshold
: Minimum value of a volume cell to contain a grid point

Properties

Distribution Method

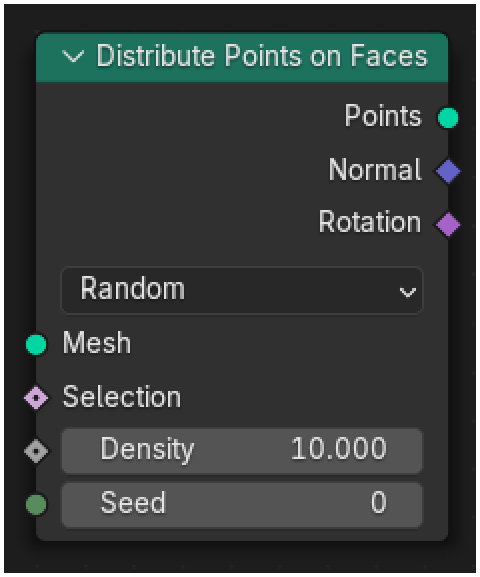
Random::	Distribute points randomly inside of the volume. The local point count is implicitly defined as a product of the global from the *Density* input and the local voxel value. This method creates a distribution that is not stable as the input volume deforms.
Grid::	Distribute the points in a grid pattern inside of the volume. At each grid point, the voxel value is used to determine whether to add a point.

Outputs

Points

Standard point cloud geometry output.

Distribute Points on Faces



The *Distribute Points on Faces* node places points on the surface of the input geometry object. Point, corner, and polygon attributes of the input geometry are transferred to the generated points. That includes vertex weights and UV maps. Additionally, the node has *Normal* and *Rotation* outputs.

The node also generates a stable ID, stored in the built-in `id` attribute, used as a stable identifier for each point. When the mesh is deformed or the density changes the values will be consistent for each remaining point. This attribute is used in the Random Value and Instance on Points nodes.

Inputs

Mesh

Standard geometry input.

Note

The input geometry must contain a mesh with faces.

Selection

The selection of which face corners should be considered for point distribution.

Distance Min

The minimal distance points can have to each other. This option is only available for the *Poisson Disk* distribution method. At its default value of zero, the node's behavior is the same as it is in *Random* mode, because none of the internally generated points are removed.

Density Max

The point density for the point distribution. The unit is in number of points per square meter. This value is multiplied by the values from the *Density* input. Only available in *Poisson Disk* mode.

Note

This will be capped on distributions by the *Distance Min* option. If the density is greater than what the minimal distance allows, no new points will be added after this threshold has been passed.

Density

The number of points to distribute per square meter on each mesh face. This value is multiplied by the values from the *Density Attribute*.

In *Poisson Disk* mode, this value is multiplied by the *Density Max* input for the final density.

Seed

The random Seed to use when generating points.

Properties

Distribution Method

Random::	Distribute points randomly on the surface. This is the fastest distribution method.
Poisson Disk::	Distribute points randomly on the surface while taking a minimum distance into account.

Legacy Normal

By default, the node uses smooth and custom normals for the *Normal* and *Rotation* values . An earlier version of this node only uses "true" normals, this option brings back this behavior of only using "true" normals.

This option can only be available in the Sidebar.

Outputs

Points

Generated points. Named attributes are copied to the result mesh, along with the data in the other attribute field outputs.

Normal

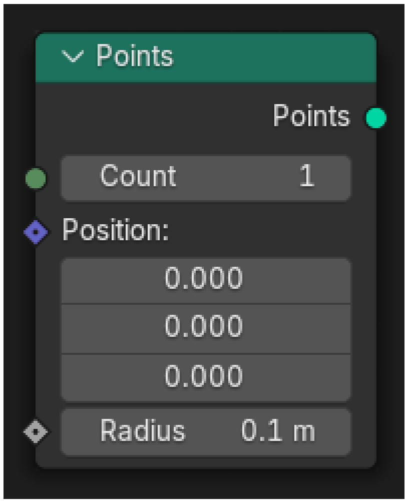
The Normal of the triangle on which each point is scattered.

Rotation

An XYZ Euler rotation built from the normal attribute for convenience. Such a value can also be built from the normal with the Euler to Rotation Node. Keep in mind that the Z axis of the result rotation will be arbitrary, since the mesh normal used to create the rotation does not have enough information to set all three rotation axes.

Points Node

The *Points* node generate a point cloud with positions and radii defined by fields.


Points
Points
Count 1
Position:
0.000
0.000
0.000
Radius 0.1 m

Inputs

Count
: The number of points to create.

Position
: The position of each generated point.

Radius
: The radius of each point.

Note

Since the point cloud is created from scratch, the *Position* and *Radius* inputs can only depend on the index node. Regular input nodes like the position won't work.

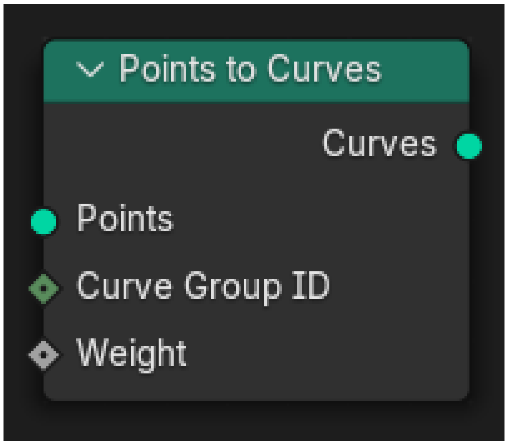
Properties

This node has no properties.

Outputs

Points
: Standard geometry output.

Points to Curves Node



The *Points to Curves* node generates a Curves geometry by taking all points and inserting them to new curves. All Attributes from points are propagated to Curve Points. Built-in curves attributes stored in points will be ignored.

Tip

To simplify thinking about points, attributes and their positions in each curve, The weight of each point in curve can be associated with a point attributes value. The sorting and grouping will be reflected on the attributes as like on the Weight and Group ID.

Inputs

Points
The Point Cloud geometry component.

Curve Group ID
All points with the same Group ID value will be joined in the same curve. The value of Group ID can be any value (negative, zero, or infinity, etc.). All created curves must have at least a single point. The order of curves depends both on Group ID value and on the order of Group ID values in the Point Cloud.

Weight
If the curve contains more than one Point, the Weight of each Point is used to define the order of all points in curve via sorting. The goal of

sorting is to have points with the minimal Weight value at the start of curve and the maximum Weight at the end of curve.

Note

If points of curve have the same Weight value, the order will be the same as its original relative location. Without any Weight and Group ID inputs, each point will have the same indices in the curve.

Properties

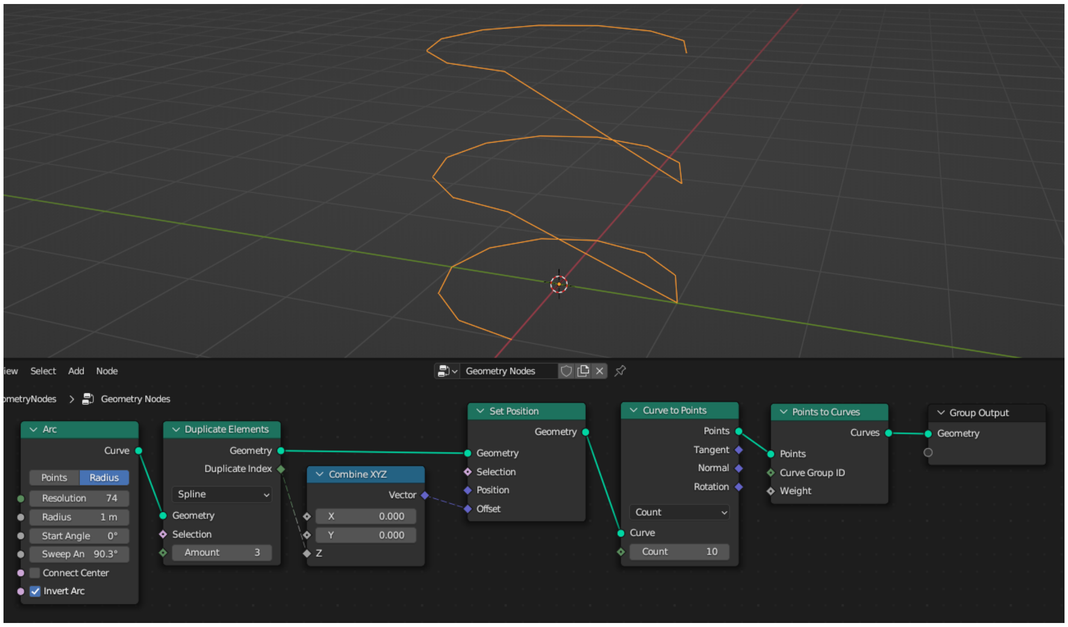
This node has no properties.

Outputs

Curves

The curves with all copied points from the Point Cloud, but joined in curves. All other components aren’t saved. The resulting curves are always non-cyclic.

Examples

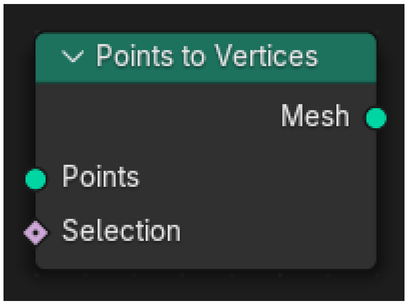

iew
Select
Add
Node
Geometry Nodes
ometryNodes
Geometry Nodes
Arc
Curve
Duplicate Elements
Geometry
Duplicate Index
Geometry
Selection
Combine XYZ
Vector
Set Position
Geometry
Geometry
Selection
Position
Offset
Curve to Points
Points
Tangent
Normal
Rotation
Curve
Points to Curves
Curves
Points
Curve Group ID
Weight
Group Output
Geometry

Points | Radius
Resolution 74
Radius 1 m
Start Angle 0°
Sweep An 90.3°
☐ Connect Center
☑ Invert Arc
Spline
Amount 3
X 0.000
Y 0.000
Z
Count
Count 10

The above example creates a curve Array with connections between curves. This is created by duplicating the Arc primitive curve with the Duplicate Elements Node. Each curve is shifted in a top direction based on its index value. All the curves are converted to the Point Cloud by the Curve to Points Node. Finally, the points are converted to curves by the Points to Curves node.

All the Points of the resulting Curves geometry have the same attributes as points on the initial Arc primitive.

Points to Vertices Node


Points to Vertices
Mesh
Points
Selection

The *Points to Vertices* node generate a mesh vertex in the output geometry for each point cloud point in the input geometry.

Inputs

Geometry

Standard geometry input.

Selection

Boolean field used to determine if each point will be converted to a vertex.

Properties

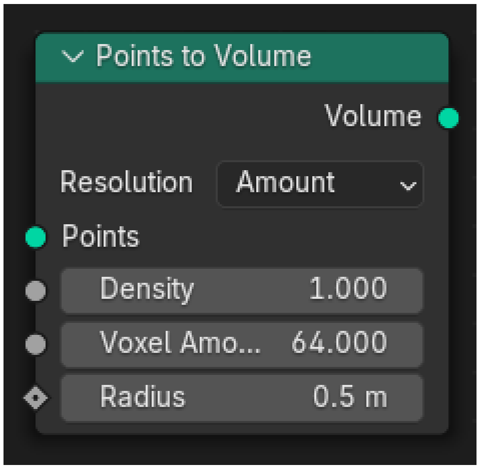
This node has no properties.

Outputs

Geometry

Standard geometry output.

Points to Volume Node



Resolution Amount
Density 1.000
Voxel Amo... 64.000
Radius 0.5 m

The *Points to Volume* node generates a fog volume sphere around every point in the input geometry. The new volume grid is named “density”.

It usually makes sense to combine this node with the Volume to Mesh Node.

Warning

This node expects that point positions are not extremely large. For position values of many billions, the behavior isn’t guaranteed, and it may be unstable.

Inputs

Points
: Standard geometry input.

Density
: Value of voxels inside the generated fog volume.

Voxel Amount
Specify the approximate number of voxels along the diagonal.

Voxel Size
Specify the voxel side length.

Radius
Specify the radius of the sphere generated at each point.

Properties

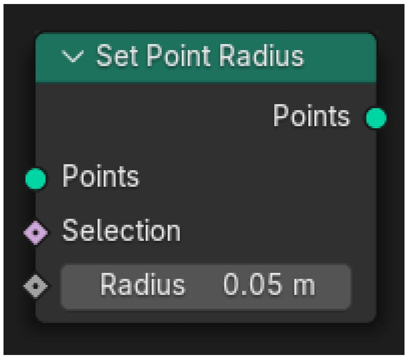
Resolution
How the voxel size is specified.

Amount::	Specify the approximate number of voxels along the diagonal.
Size::	Specify the voxel side length. It is recommended to be careful when tweaking this value, because small changes can have a large effect on the processing time.

Outputs

Volume
Standard geometry output.

Set Point Radius Node


Set Point Radius
Points
Points
Selection
Radius 0.05 m

The *Set Point Radius* node controls the size each selected point cloud point should display with in the viewport.

The input node for this data is the Radius Node.

Inputs

Geometry
: Standard geometry input.

Radius
: Float value indicating the radius of the point geometry at each point.

Selection
: Boolean input for selecting which points will have the radius value applied.

Properties

This node has no properties.

Outputs

Geometry
: Standard geometry output.

Volume Nodes

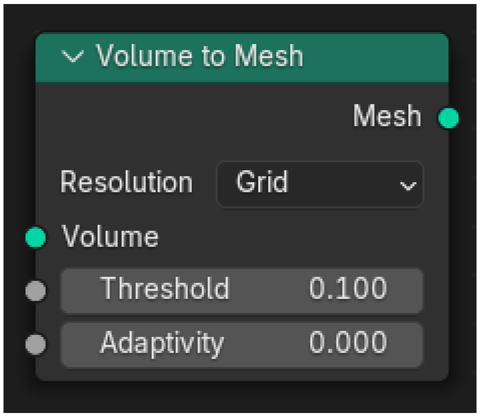
Nodes for creating or working with volumes.

- Operations
- Primitives

Volume Operation Nodes

- Volume to Mesh Node

Volume to Mesh Node



Resolution Grid

Threshold 0.100

Adaptivity 0.000

The *Volume to Mesh* node generates a mesh on the “surface” of a volume. The surface is defined by a threshold value. All voxels with a larger value than the threshold are considered to be inside.

Inputs

Volume
: Standard geometry input.

Voxel Amount
: Specifies the approximate resolution of the final mesh. The voxel size is adapted to the size of the entire volume.

Voxel Size
: Use a fixed resolution that does not change when the volume changes.

Threshold
: Voxels with a larger value are considered to be inside the mesh. The mesh will be generated on the boundary of inside and outside voxels. This is also called “iso value”.

Adaptivity

Reduces the final face count by simplifying geometry where detail is not needed. This is similar to decimating the final to reduce resolution where it is not needed.

Properties

Resolution Mode

Mode for how the resolution of the final mesh is controlled.

Grid::	This makes the resolution dependent on the resolution of the grid that is converted. Higher resolution grids result in a higher resolution mesh. In many cases, that is the most efficient mode.
Voxel Amount::	Specifies the approximate resolution of the final mesh. The voxel size is adapted to the size of the entire volume.
Voxel Size::	Use a fixed resolution that does not change when the volume changes.

Note

This option applies individually for every grid in the input geometry.

Outputs

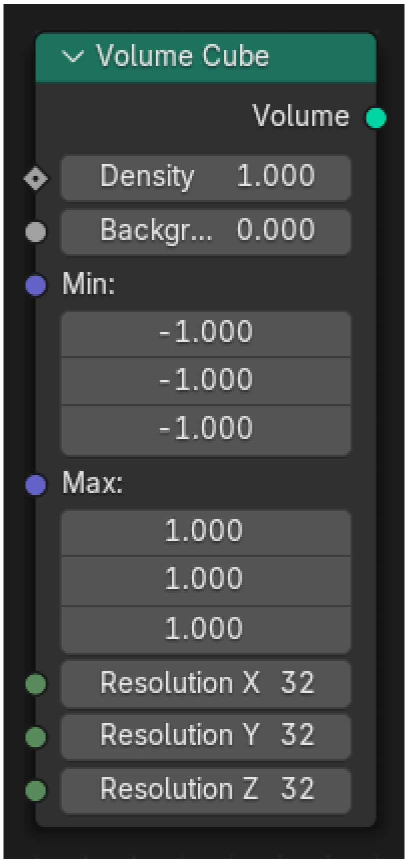
Mesh

Standard geometry output.

Volume Primitive Nodes

- Volume Cube Node

Volume Cube Node


Volume Cube
Volume
Density 1.000
Backgr... 0.000
Min:
-1.000
-1.000
-1.000
Max:
1.000
1.000
1.000
Resolution X 32
Resolution Y 32
Resolution Z 32

The *Volume Cube* generates a volume from scratch by evaluating an input field on every single voxel in a rectangular prism. The *Density* field defines the output volume grid's value at every voxel. The field can only depend on the Position Node.

Inputs

Density

The value for the new grid at each voxel.

Background

The value of the grid outside the rectangular prism controlled by the *Min* and *Max* inputs. The node can generate a more memory-efficient volume when the values of the *Density* input are the same as the background value.

Min

One corner of the rectangular prism in which to fill evaluate the field.

Max
: The other corner of the rectangular prism in which to fill evaluate the field.

Resolution X,Y,Z
: The number of voxels to evaluate the field in on each axis.

> **Note**
>
> Changing these values can have a significant impact on performance. For example, the default values of 32 mean the input field will be evaluated about 33 thousand times. Increasing the values to 100 will give 1 million evaluations, and 1000 would give 1 billion.

Properties

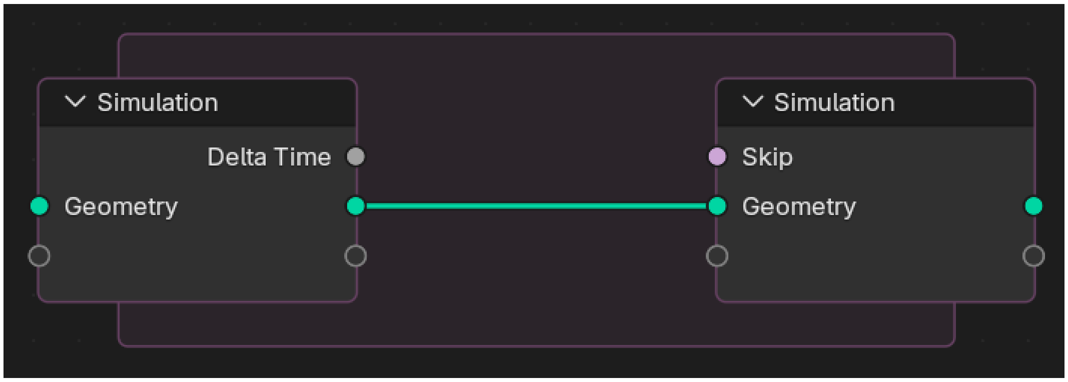
This node has no properties.

Outputs

Volume
: Geometry containing the generated volume.

Simulation Zone

Simulation zones allow the result of one frame to influence the next one. That way even a set of simple rules can lead to complex results, with the passing of time. The most common type of them is physics simulation, with specific solvers for physical phenomena.



Initial simulation nodes and simulation zone.

When adding a simulation, two nodes are added, defining between them a “Simulation Zone”.

The inputs that are connected to the Simulation Input node are evaluated only once, at the beginning of the simulation, passed to the next simulation state and eventually outputted. Other nodes can be linked inside the simulation region from the outside. Those are re-evaluated every step based on their value at the given frame.

It is not possible to have any link going towards outside. The result of the simulation can only be accessed via the Simulation Output node. This also allows sub-frame interpolation for motion blur.

Note

This node cannot be used in the Tool context—only in the *Modifier* context.

Clock

The simulation is tied to the animation system, with support for sub-steps. It will only be evaluated while the animation frame changes, and is cached like the existing physics simulations in Blender.

Properties

In the Node Editor the inputs can be renamed, shuffled and removed. This is also the place where sub-steps can be defined for a simulation.

Inputs

Geometry

Standard geometry input, which is available by default to input geometry into the simulation zone. More bake items can be added by dragging sockets into the blank socket or in the *Simulation State* panel. Items can be renamed by `Ctrl-LMB` on the socket name or in the nodes *Properties* panel.

Delta Time

The time in seconds between frames. Essentially this the inverse of the render Frame Rate.

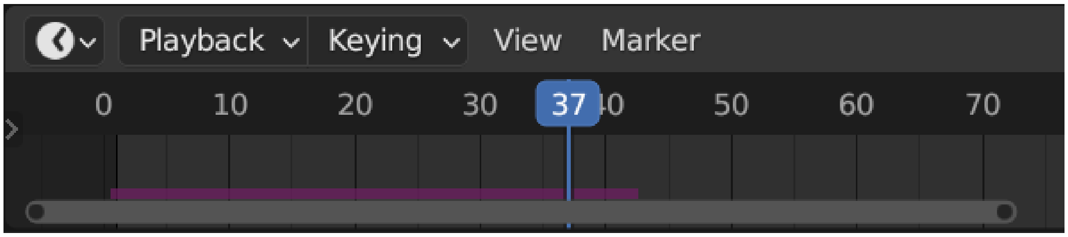
This delta is used to drive the simulation by connecting it node setups that depend on a rate. This will keep the simulation playback consistent when the frame rate changes.

Skip

Forward the output of the simulation input node directly to the output node and ignore the nodes in the simulation zone.

Baking

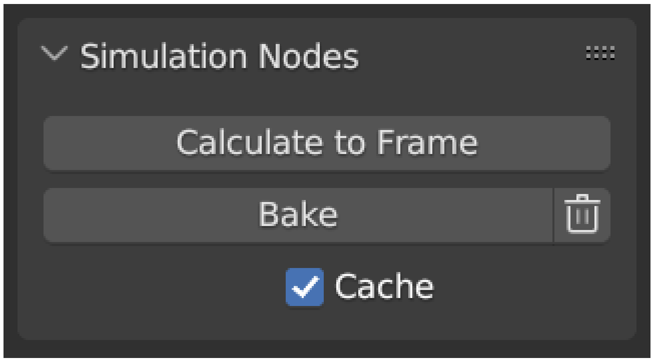
The simulation is automatically cached during playback. The valid cache can be seen as a strong yellow line in the timeline editor. This allows for animators to quickly inspect all the previous frames of a simulation.


Playback
Keying
View
Marker
0
10
20
30
37
40
50
60
70

Cached frames in the Timeline.

For the cases where the current frame is the only one relevant, users can opt-out of “Cache” to save memory.

When the result is ready to be sent to a render-farm, it can be baked to disk. This allows for the simulation to be rendered in a non-sequential order.


Simulation Nodes
Calculate to Frame
Bake
☑ Cache

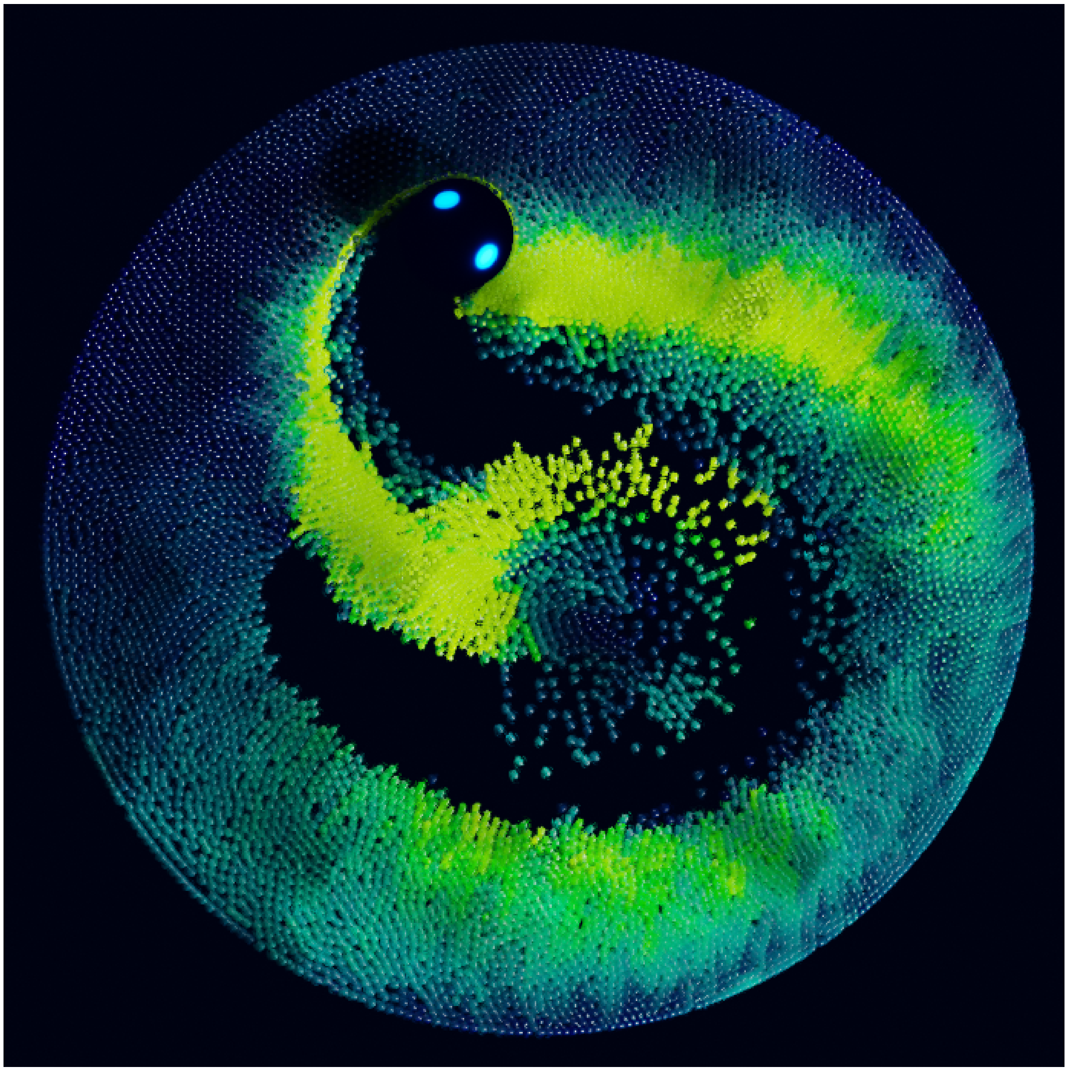
Simulation and Physics, Simulation Nodes user interface.

Note

Baking the simulation will bake all the simulations in all modifiers for the selected objects.

Examples

Combined with the Index of Nearest, this can be used for a number of sphere-based simulations.



Index of Nearest sample file CC-BY Sean Christofferson.

Material Nodes

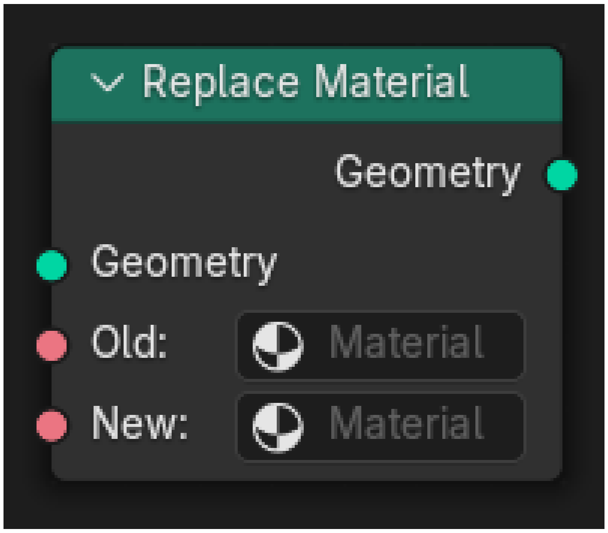
Nodes that work with materials.

- Replace Material Node

- Material Index Node
- Material Selection Node

- Set Material Node
- Set Material Index Node

Replace Material Node


Replace Material
Geometry
Geometry
Old:
Material
New:
Material

The *Replace Material* node swaps one material with another. Replacing a material with this node is more efficient than creating a selection of all faces with the old material with the Material Selection Node and then using the Set Material Node.

Note

Currently this node only adjusts mesh data.

Inputs

Geometry
: Standard geometry input.

Old
: Material that is going to be replaced.

New
: Material that is replacing the old material.

Properties

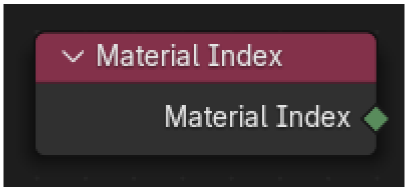
This node has no properties.

Outputs

Geometry

Standard geometry output.

Material Index Node



The *Material Index* node outputs which material in the list of materials of the geometry each element corresponds to. Currently the node supports mesh data, where `material_index` is a built-in attribute on faces.

The node to set this data is the Set Material Index node.

Inputs

This node has no inputs.

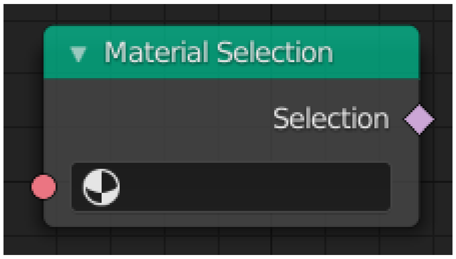
Properties

This node has no properties.

Outputs

Material Index
    Standard integer value, with a minimum value of zero.

Material Selection Node



The *Material Selection* node provides a selection for meshes that use this material. Since the `material_index` is stored on each face, the output will be implicitly interpolated to a different domain when necessary. For example, every vertex connected to a selected face will be selected.

Inputs

Material
    Standard material input.

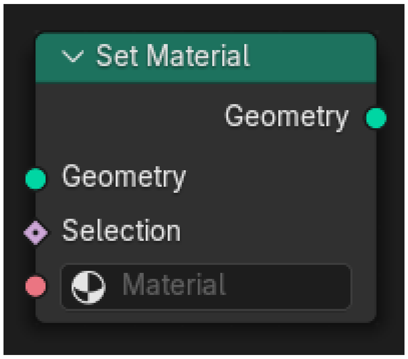
Properties

This node has no properties.

Outputs

Selection
    Selection of faces that use the input material.

Set Material Node



The *Set Material* changes the material assignment in the specified selection, by adjusting the `material_index` attribute. If the material is already used on the geometry, the existing material index will be reused.

Note

This node adjusts mesh, point clouds, and volume data; other data types do not support materials.

Inputs

Geometry

Standard geometry input containing a mesh.

Material

The material to apply to the geometry.

Selection

Whether to change the material of each face. True values mean the material will be changed, false values mean it will remain the same.

Note, volumes and point clouds only support a single material, in these cases a field input will be ignored.

Properties

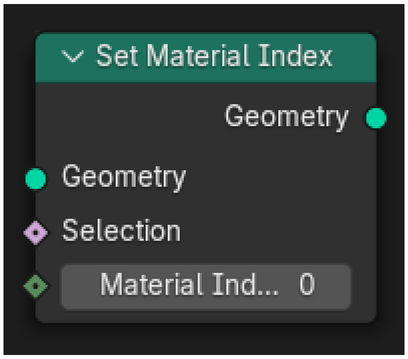
This node has no properties.

Outputs

Geometry

Standard geometry output.

Set Material Index Node


Set Material Index
Geometry
Geometry
Selection
Material Ind... 0

The *Set Material Index* node sets the material index for a geometry.

The node to get this data is the Material Index node.

Inputs

Geometry
: Standard geometry input.

Selection
: Whether to change the material index for each face. True values mean the material index will be changed, false values mean it will remain the same.

Material Index
: The new material index.

Properties

This node has no properties.

Outputs

Geometry
: Standard geometry output.

Texture Nodes

Nodes to add textures.

Tip

Texture nodes can produce details at a higher frequency than geometry can show. This is more evident with textures that produce abrupt changes such as brick and checker. This may cause artifacts such as Moiré type patterns or a lack of detail due to insufficient sampling points.

- Brick Texture Node
- Checker Texture Node
- Gabor Texture Node
- Gradient Texture Node
- Image Texture Node
- Magic Texture Node
- Musgrave Texture Node
- Noise Texture Node
- Voronoi Texture Node
- Wave Texture Node
- White Noise Texture Node

Brick Texture Node

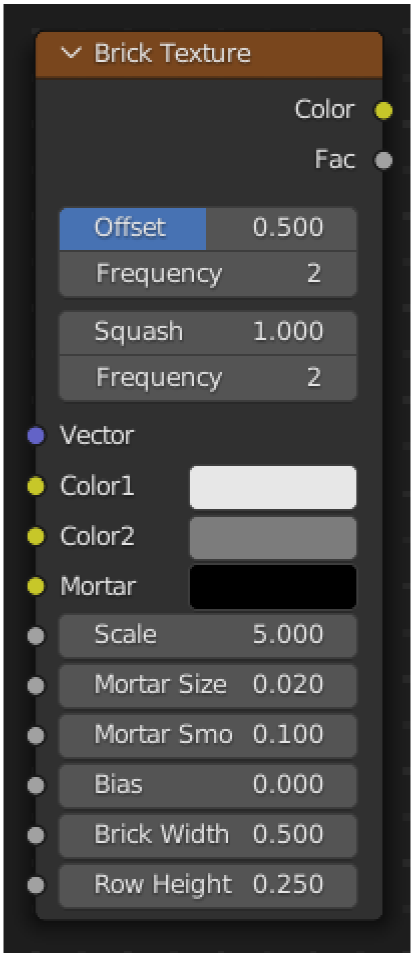
Note

This node is ported from shader nodes. The manual and images are referencing the shader version of the node. This node accepts field inputs and outputs. When not connected the Vector input has an implicit `position` attribute value.

Tip

Texture nodes can produce details at a higher frequency than geometry can show. This may cause artifacts such as Moiré type patterns or a lack of detail due to insufficient sampling points.

The *Brick Texture* is used to add a procedural texture producing bricks.


Brick Texture
Color
Fac
Vector
Color1
Color2
Mortar

Offset 0.500
Frequency 2
Squash 1.000
Frequency 2
Scale 5.000
Mortar Size 0.020
Mortar Smo 0.100
Bias 0.000
Brick Width 0.500
Row Height 0.250

Inputs

Color 1/2
: Color of the bricks.

Mortar
: The color of the area between bricks.

Scale
: Overall texture scale.

Mortar Size
: The size of the filling between the bricks known as “mortar”; 0 means no mortar.

Mortar Smooth

Blurs/softens the edge between the mortar and the bricks. This can be useful with a texture and displacement textures.

Bias

The color variation between *Color 1/2*. Values of -1 and 1 only use one of the two colors; values in between mix the colors.

Brick Width

The width of the bricks.

Row Height

The height of the brick rows.

Properties

Offset

Determines the brick offset of the various rows.

Frequency

Determines the offset frequency. A value of 2 gives an even/uneven pattern of rows.

Squash

Amount of brick squashing.

Frequency

Brick squashing frequency.

Outputs

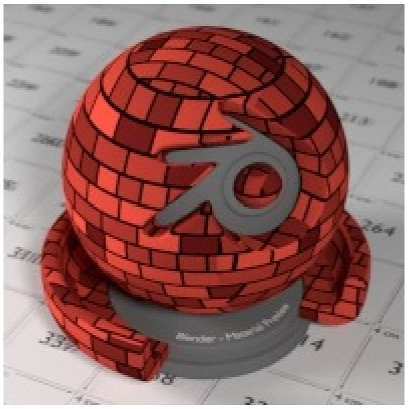
Color

Texture color output.

Factor

Mortar mask (1 = mortar).

Examples



Brick texture: Colors changed, Squash 0.62, Squash Frequency 3.

Checker Texture Node

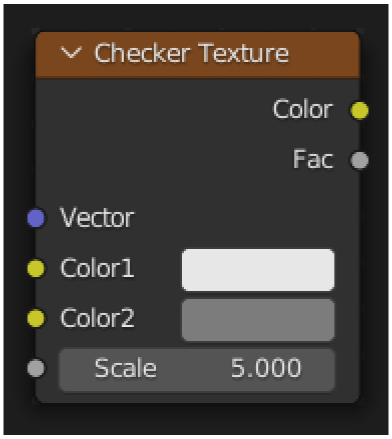
Note

This node is ported from shader nodes. The manual and images are referencing the shader version of the node. This node accepts field inputs and outputs. When not connected the Vector input has an implicit `position` attribute value.

Tip

Texture nodes can produce details at a higher frequency than geometry can show. This may cause artifacts such as Moiré type patterns or a lack of detail due to insufficient sampling points.

The *Checker Texture* is used to add a checkerboard texture.



Inputs

Vector

Texture coordinate to sample texture at; defaults to Generated texture coordinates if the socket is left unconnected.

Warning

> This node can have precision issues with some vector inputs. See the notes for the White Noise Texture for ways to mitigate this issue.

Color1, Color 2
: Color of the checkers.

Scale
: Overall texture scale. The scale is a factor of the bounding box of the face divided by the scale. For example, a scale of 15 will result in 15 alternate patterns over the overall UV bounding box. Different patterns could be achieved using other nodes to give different input patterns to this socket. For example, using the Math node.

Properties

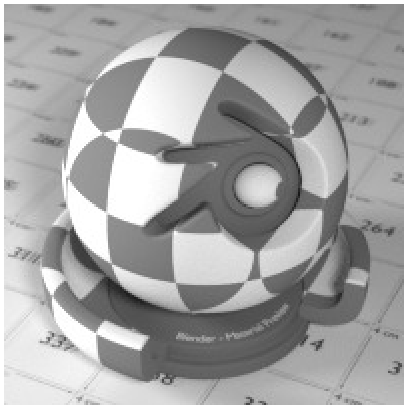
This node has no properties.

Outputs

Color
: Texture color output.

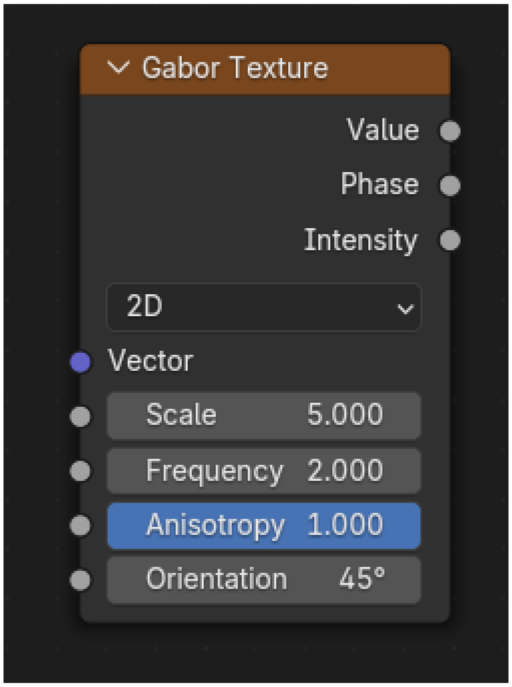
Factor
: Checker 1 mask (1 = Checker 1).

Examples



Default Checker texture.

Gabor Texture Node



2D

Scale 5.000

Frequency 2.000

Anisotropy 1.000

Orientation 45°

The *Gabor Texture* node evaluates a Gabor noise at the input texture coordinates. Gabor noise is visually characterized by random interleaved bands whose direction and width can be controlled. Additionally, it can be used to create omnidirectional noise like the standard Noise Texture node, but since it is more expensive to compute, using the Noise Texture node is probably the better option in those cases. See the examples for more information.

Note

This node is ported from shader nodes.
The manual and images are referencing the shader version of the node.
This node accepts field inputs and outputs. When not connected the Vector input has an implicit `position` attribute value.

Inputs

Vector
: The coordinates at which Gabor noise will be evaluated. The Z component is ignored in the 2D case. Defaults to *Generated* texture coordinates if the socket is left unconnected.

Scale
: Scale of the Gabor noise.

Frequency

The rate at which the Gabor noise changes across space. This is different from the Scale input in that it only scales perpendicular to the Gabor noise direction.

Anisotropy

The directionality of Gabor noise. 1 means the noise is completely directional, while 0 means the noise is omnidirectional.

Orientation

The direction of anisotropic Gabor noise. This is an angle for the 2D case, while it is a unit direction vector for the 3D case.

Properties

Type

Type of Gabor noise texture.

2D::	Evaluates the noise in 2D space. The Z component of the input vector is ignored.
3D::	Evaluates the noise in 3D space.

Note

Higher dimensions corresponds to higher render time, so lower dimensions should be used unless higher dimensions are necessary.

Outputs

Value

The Gabor noise value with both random intensity and phase. This is equal to sine the phase multiplied by the intensity.

Phase

The phase of the Gabor noise, which has no random intensity.

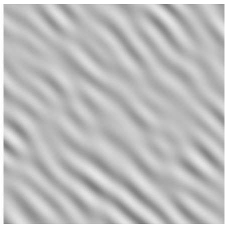
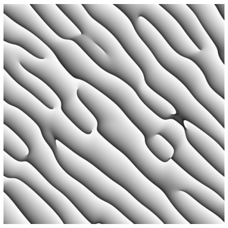
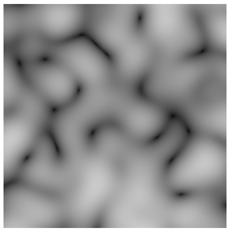
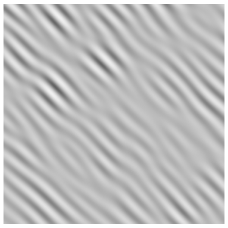
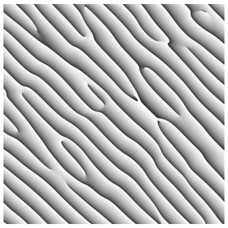
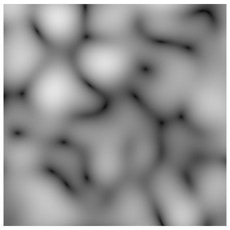
Intensity

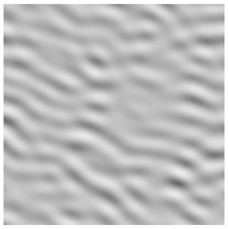
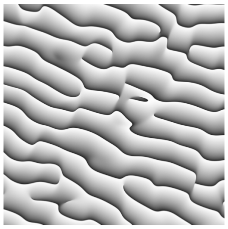
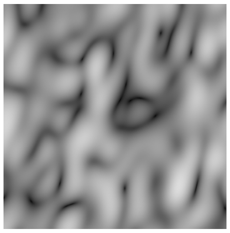
The intensity of the Gabor noise, which has no random phase.

Examples

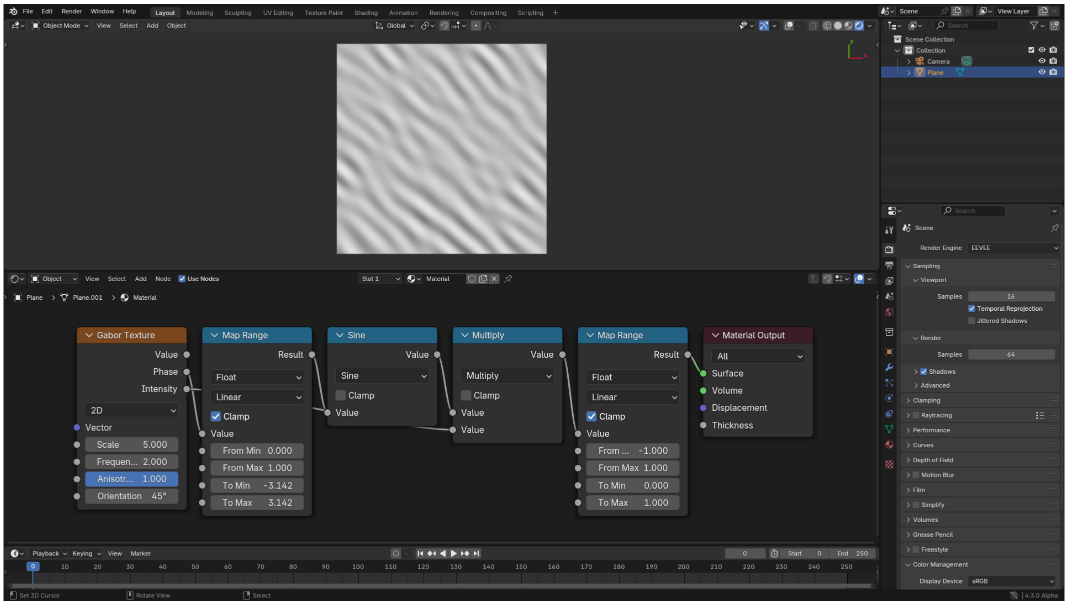
The following table demonstrates different outputs of the node with different parameters. As can be seen, the noise is visually characterized by interleaved bands that are generally oriented in a specific direction. But the *Anisotropy* parameter can be decreased below 1 to make the bands more random in directions. The *Frequency* parameter determines the number of bands perpendicular to the direction of the noise. However, the *Scale* parameter can also be used to globally increase the number of bands, so consider increasing the scale first since high frequency noise can suffer from low contrast and limited interleaving of bands.

Different outputs with different parameters.

 Value output. Frequency = 2. Anisotropy = 1.	 Phase output. Frequency = 2. Anisotropy = 1.	 Intensity output. Frequency = 2. Anisotropy = 1.
 Value output. Frequency =	 Phase output. Frequency =	 Intensity output.

3. Anisotropy = 1.	3. Anisotropy = 1.	Frequency = 3. Anisotropy = 1.
 Value output. Frequency = 2. Anisotropy = 0.7.	 Phase output. Frequency = 2. Anisotropy = 0.7.	 Intensity output. Frequency = 2. Anisotropy = 0.7.

Gabor noise is decomposed into a *Phase* and an *Intensity* components, where the Gabor value is computed as sine the phase multiplied by the intensity, noting that the phase output is normalized to the [0, 1] range.



Gabor Texture: 2D, Scale 5.000, Frequency 2.000, Anisotropy 1.000, Orientation 45°

Map Range: Float, Linear, Clamp ☑, From Min 0.000, From Max 1.000, To Min -3.142, To Max 3.142

Sine: Sine, Clamp ☐

Multiply: Multiply, Clamp ☐

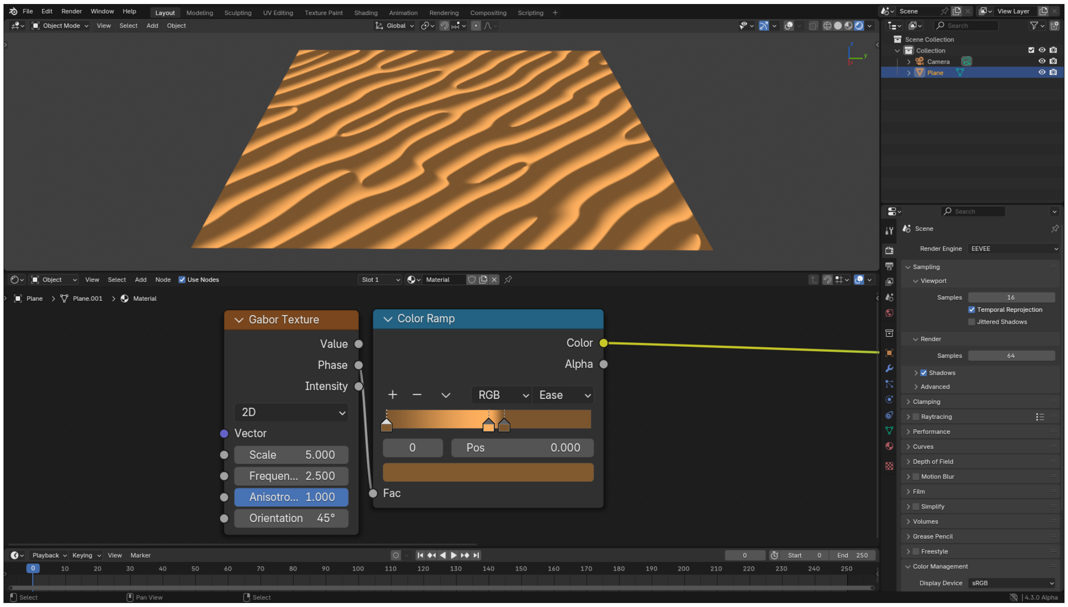
Map Range: Float, Linear, Clamp ☑, From Min -1.000, From Max 1.000, To Min 0.000, To Max 1.000

Material Output: All

Object
View
Select
Add
Node
Use Nodes
Slot 1
Material
Plane
Plane.001
Material
Value
Phase
Intensity
Vector
Result
Value
Surface
Volume
Displacement
Thickness
Playback
Keying
View
Marker
Start 0
End 250
Rotate View
Select
Render Engine EEVEE
Sampling
Viewport
Samples 16
Temporal Reprojection
Jittered Shadows
Render
Samples 64
Shadows
Advanced
Clamping
Raytracing
Performance
Curves
Depth of Field
Motion Blur
Film
Simplify
Volumes
Grease Pencil
Freestyle
Color Management
Display Device sRGB
Scene Collection
Collection
Camera
Plane
4.3.0 Alpha

Compute the value output from the phase and intensity outputs.

The advantage of the *Phase* output is that it has no random intensities and no low contrast areas as in the value output, so it can be used as a base for textures that are more structured in nature, like sand dunes.

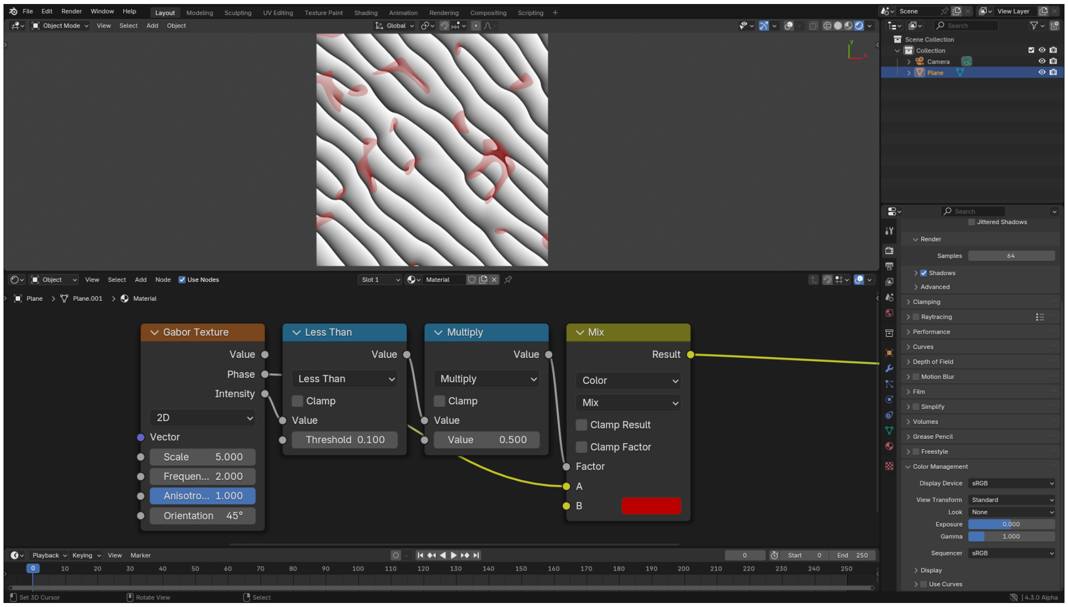

Gabor Texture
Value
Phase
Intensity
2D
Vector
Scale 5.000
Frequen... 2.500
Anisotro... 1.000
Orientation 45°

Color Ramp
Color
Alpha
RGB
Ease
0
Pos 0.000
Fac

Render Engine EEVEE
Samples 16
☑ Temporal Reprojection
☐ Jittered Shadows
Samples 64
☑ Shadows
Display Device sRGB
File
Edit
Render
Window
Help
Layout
Modeling
Sculpting
UV Editing
Texture Paint
Shading
Animation
Rendering
Compositing
Scripting
Object Mode
View
Select
Add
Object
Global
Scene
View Layer
Scene Collection
Collection
Camera
Plane
Node
Use Nodes
Slot 1
Material
Plane.001
Scene
Sampling
Viewport
Render
Advanced
Clamping
Raytracing
Performance
Curves
Depth of Field
Motion Blur
Film
Simplify
Volumes
Grease Pencil
Freestyle
Color Management
Playback
Keying
Marker
Start 0
End 250
Pan View
4.3.0 Alpha

Sand dune-like structures creates using the phase output.

The main advantage and use of the *Intensity* output is that it provides information about the location of *singularities* in the *Phase* output. Singularities are those areas in the phase where the bands meet, which are shown in red in the following figure. Those areas will be close to zero in the *Intensity* output. So if those areas are undesirable, they can be hidden by multiplying by a variant of the *Intensity* output.



Gabor Texture
2D
Scale 5.000
Frequen... 2.000
Anisotro... 1.000
Orientation 45°

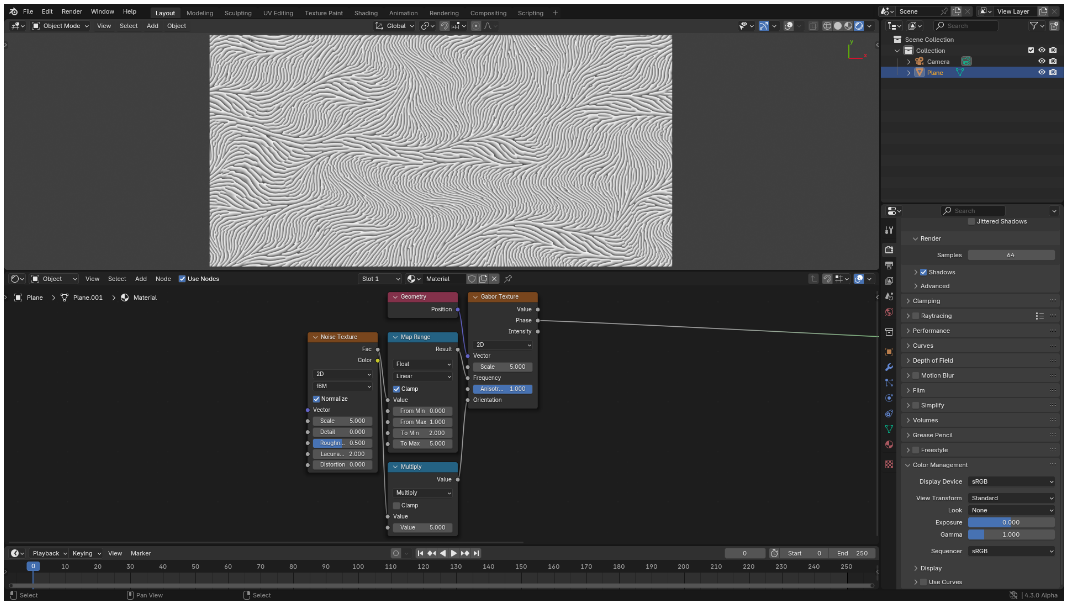
Less Than
Less Than
☐ Clamp
Threshold 0.100

Multiply
Multiply
☐ Clamp
Value 0.500

Mix
Color
Mix
☐ Clamp Result
☐ Clamp Factor

Visualization of the areas where singularities happen.

Inputs can be varies across space to get more interesting patterns.



Noise Texture
2D
fBM
☑ Normalize
Scale 5.000
Detail 0.000
Roughn... 0.500
Lacuna... 2.000
Distortion 0.000

Geometry
Position

Map Range
Float
Linear
☑ Clamp
From Min 0.000
From Max 1.000
To Min 2.000
To Max 5.000

Multiply
Multiply
☐ Clamp
Value 5.000

Gabor Texture
2D
Scale 5.000
Anisotr... 1.000

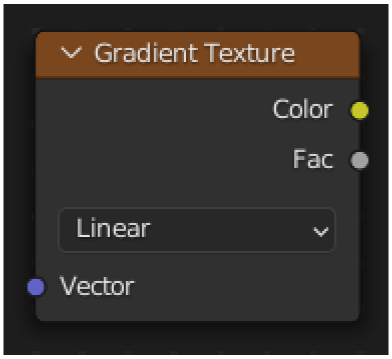
Varying the frequency and orientation across space.

Gradient Texture Node

Note

This node is ported from shader nodes. The manual and images are referencing the shader version of the node. This node accepts field inputs and outputs. When not connected the Vector input has an implicit `position` attribute value.

The *Gradient Texture* node generates interpolated color and intensity values based on the input vector.



Inputs

Vector
: Texture coordinate to sample texture at; defaults to Generated texture coordinates if the socket is left unconnected.

Properties

Type
: Controls the type of gradient generated.

Linear::	Directly outputs the input X coordinate.
Quadratic::	Interpolates the input X coordinate quadratically.
Easing::	

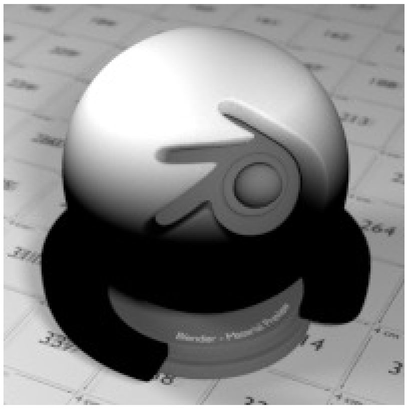
	Uses a combination of quadratic and linear interpolation to generate a smooth gradient from the input X coordinate.
Diagonal::	Averages the input X and Y coordinates.
Spherical::	Creates an inverse gradient using the length of the input vector; the maximum value is at (0, 0, 0).
Quadratic Sphere::	The same as Spherical, except interpolated quadratically.
Radial::	Outputs a value based on the angle of the input around the Z axis.

Outputs

Color
Texture color output.

Factor
Texture intensity output.

Examples



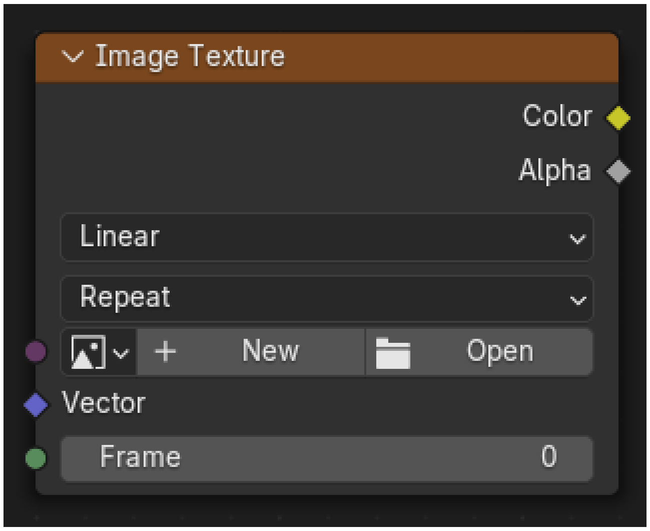
Gradient texture using object coordinates.

Image Texture Node

Note

Unlike the other texture nodes, this node operates differently in geometry nodes compared to the equivalent shader node. When not connected the Vector input has an implicit `position` attribute value.

The *Image Texture* node is used to add an image file as a texture. The image data is sampled with the input Vector and outputs a Color and Alpha value.



Inputs

Image
: The image socket can be used to connect to the Group Input node. If this is not connected the image can be opened or selected from the node.

Vector
: Texture coordinate for texture look-up. If this socket is left unconnected, the Position attribute is used.

Frame
: If the Image supports animation, the frame can be set here. This can be keyframed so that the image changes between frames.

Properties

Interpolation

Method to scale images up or down for sampling.

Linear::	Regular quality interpolation.
Cubic::	Smoother, better quality interpolation. For bump maps this should be used to get best results.
Closest::	No interpolation, use only closest pixel for rendering pixel art.

Extension

Extension defines how the image is extrapolated past the original bounds:

Repeat::	Will repeat the image horizontally and vertically giving tiled-looking result.
Extend::	Will extend the image by repeating pixels on its edges.
Clip::	Clip to the original image size and set all the exterior pixels values to transparent black.
Mirror::	Repeatedly flip the image horizontally and vertically.

Outputs

Color

RGBA color from the image.

Alpha

Alpha channel from image.

Examples

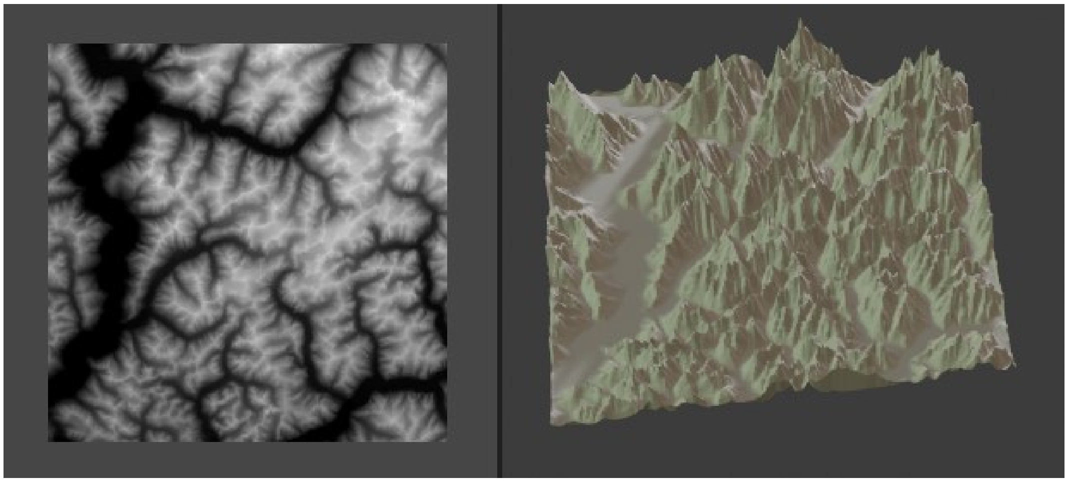


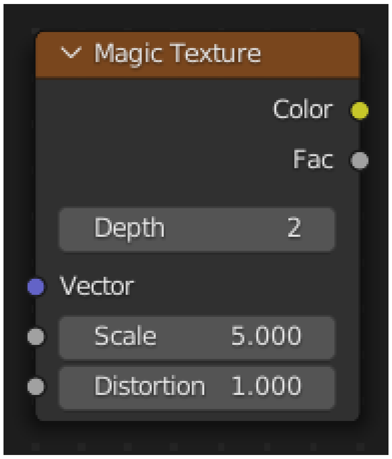
Image Texture displacing a plane.

Magic Texture Node

Note

This node is ported from shader nodes. The manual and images are referencing the shader version of the node. This node accepts field inputs and outputs. When not connected the Vector input has an implicit `position` attribute value.

The *Magic Texture* node is used to add a psychedelic color texture.


Magic Texture
Color
Fac
Depth 2
Vector
Scale 5.000
Distortion 1.000

Inputs

Vector
: Texture coordinate to sample texture at; defaults to Generated texture coordinates if the socket is left unconnected.

Scale
: Scale of the texture.

Distortion
: Amount of distortion.

Properties

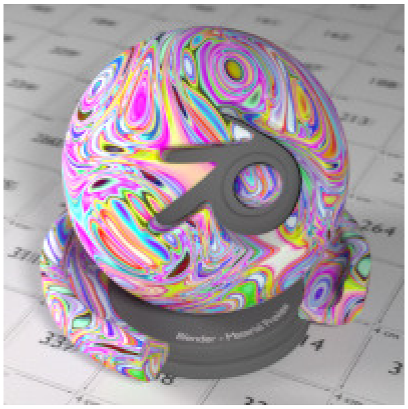
Depth
: Number of iterations.

Outputs

Color
: Texture color output.

Factor
: Texture intensity output.

Examples



Magic texture: Depth 10, Distortion 2.0.

Musgrave Texture Node

The Musgrave texture node was replaced by the Noise Texture node, which includes all the same functionality.

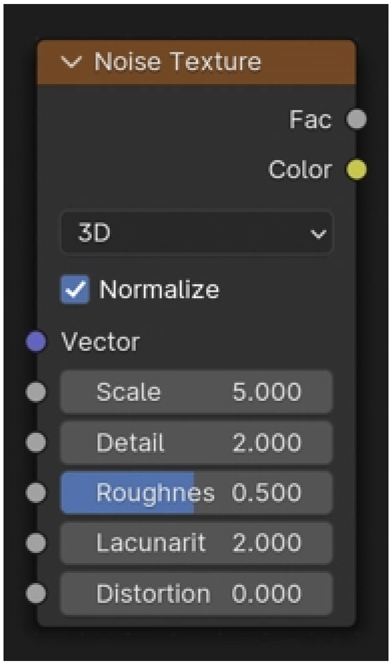
- The Dimension input was replaced by a Roughness input, where $Roughness = Lacunarity^{-Dimension}$.
- The Detail input value must be subtracted by 1 compared to the old Musgrave Texture node.

Noise Texture Node

Note

This node is ported from shader nodes. The manual and images are referencing the shader version of the node. This node accepts field inputs and outputs. When not connected the Vector input has an implicit `position` attribute value.

The *Noise Texture* node evaluates a fractal Perlin noise at the input texture coordinates. It can be used for a single Perlin noise evaluation, or for combining multiple octaves (layers) with increasingly finer detail.



3D

☑ Normalize

Scale 5.000

Detail 2.000

Roughnes 0.500

Lacunarit 2.000

Distortion 0.000

Inputs

The inputs are dynamic, they become available if needed depending on the node properties.

Vector
: Texture coordinate to evaluate the noise at; defaults to *Generated* texture coordinates if the socket is left unconnected.

W
Texture coordinate to evaluate the noise at.

Scale
Scale of the base noise octave.

Detail
Number of noise octaves. The fractional part of the input is multiplied by the magnitude of the highest octave. Higher number of octaves corresponds to a higher render time.

Roughness
Blend between a smoother noise pattern, and rougher with sharper peaks.

Lacunarity
The difference between the scale of each two consecutive octaves. Larger values corresponds to larger scale for higher octaves.

Offset
An added offset to each octave, determines the level where the highest octave will appear.

Gain
An extra multiplier to tune the magnitude of octaves.

Distortion
Amount of distortion.

Properties

Dimensions
The dimensions of the space to evaluate the noise in.

1D::	Evaluate the noise in 1D space at the input *W*.
2D::	Evaluate the noise in 2D space at the input *Vector*. The Z component is ignored.

3D::	Evaluate the noise in 3D space at the input *Vector*.
4D::	Evaluate the noise in 4D space at the input *Vector* and the input *W* as the fourth dimension.

Note

Higher dimensions corresponds to higher render time, so lower dimensions should be used unless higher dimensions are necessary.

Normalize

If enabled, ensures that the output values stay in the range 0.0 to 1.0. When disabled, output values are in the range -1.0 to 1.0.

Type

Type of Noise texture, with different ways to combine octaves.

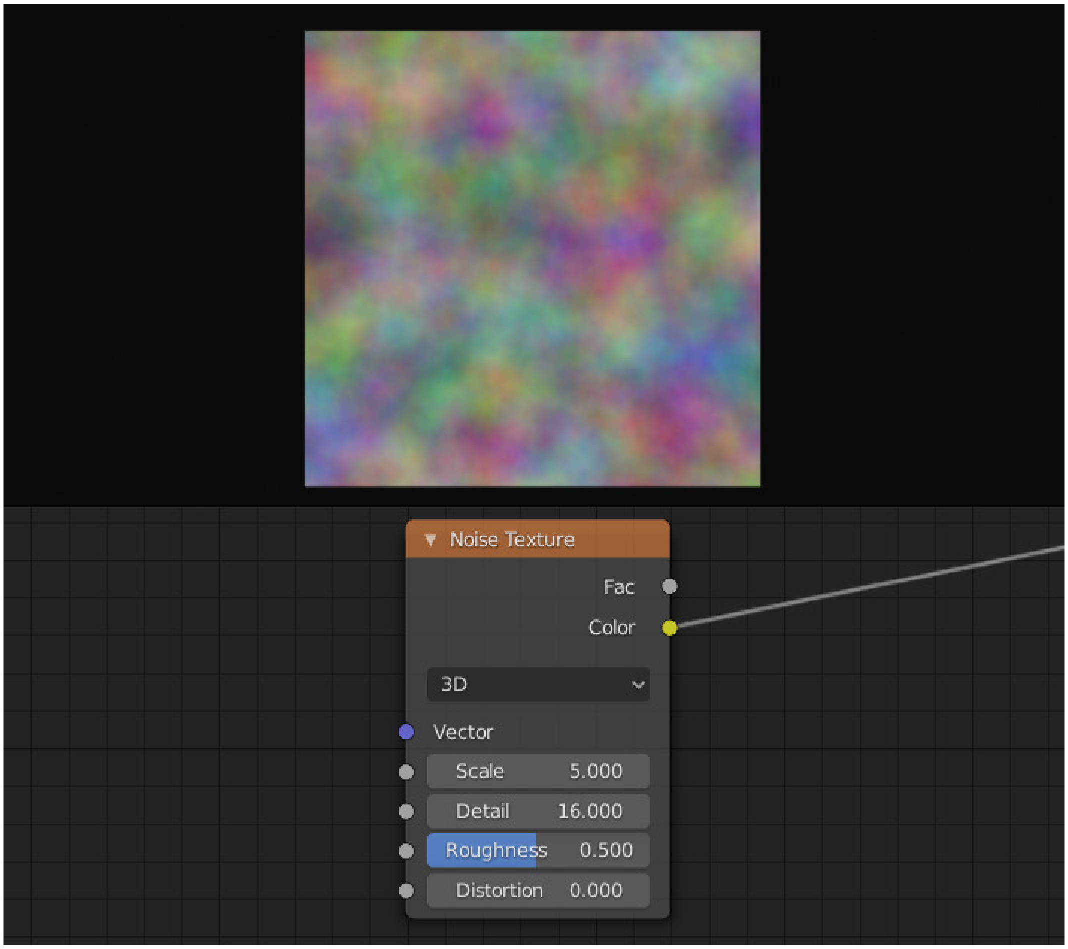
FBM::	Fractal Brownian motion, produces a homogeneous and isotropic result. Values from octaves are added together.
Multifractal::	More uneven, varying by location similar to real terrain. Values from octaves are multiplied together.
Hybrid Multifractal::	Creates peaks and valleys with different roughness values, like real mountains rise out of flat plains. Combines octaves using both addition and multiplication.
Ridged Multifractal::	Creates sharp peaks. Calculates the absolute value of the noise, creating “canyons”, and then flips the surface upside down.
Hetero Terrain::	Similar to *Hybrid Multifractal* creates a heterogeneous terrain, but with the likeness of river channels.

Outputs

Factor
: Value of fractal noise.

Color
: Color with different fractal noise in each component.

Examples


Noise Texture
Fac
Color
Vector

3D

Scale 5.000

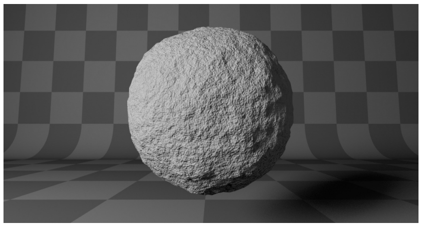
Detail 16.000

Roughness 0.500

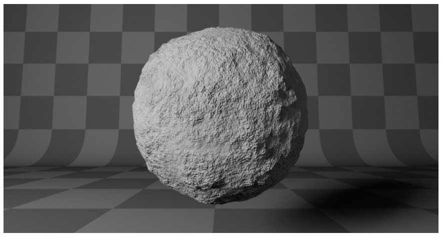
Distortion 0.000

Noise Texture with high detail.

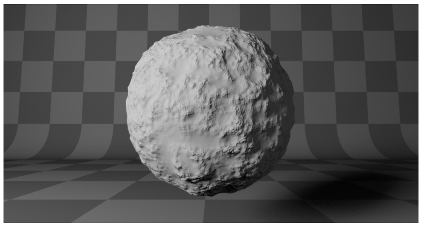
Different Noise types with the same parameters.



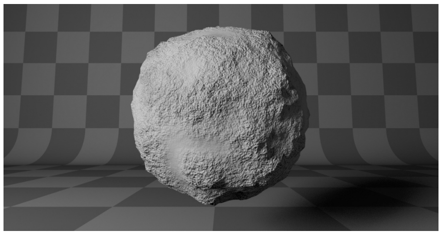
fBM (fractal Brownian Motion).



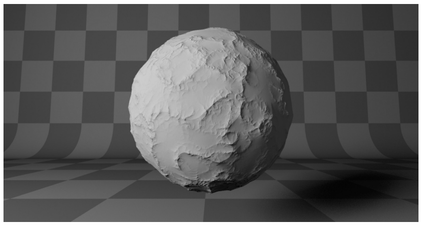
Multifractal.



Hybrid Multifractal.



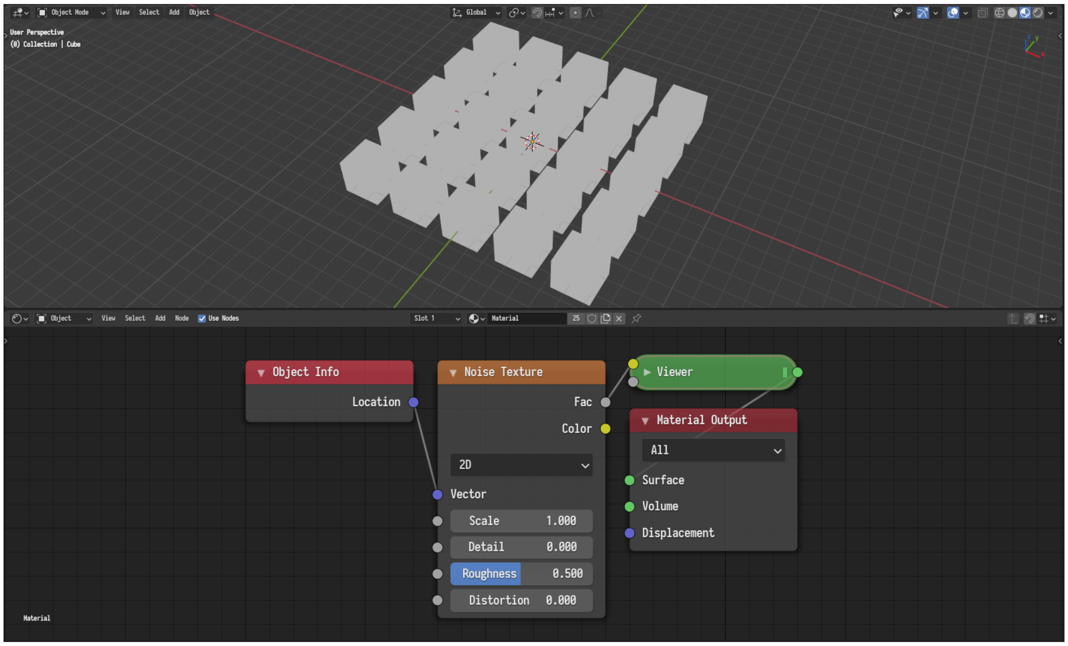
Heterogeneous Terrain.



Ridged Multifractal.

Notes

While the noise is random in nature, it follows a certain pattern that might not evaluate to random values in some configurations. For instance, consider the following configuration where a grid of objects have a material that evaluates a noise texture at their locations. One might expect the objects to have random values since they have different locations, but this is not the case.


Object Mode
View
Select
Add
Object
Global
User Perspective
(0) Collection | Cube
Object
View
Select
Add
Node
Use Nodes
Slot 1
Material
Object Info
Location
Noise Texture
Fac
Color
Vector
Viewer
Material Output
All
Surface
Volume
Displacement
Material

2D

Scale 1.000

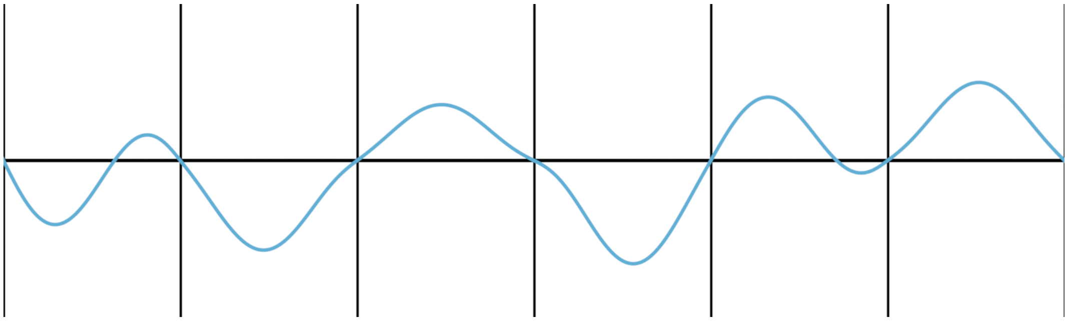
Detail 0.000

Roughness 0.500

Distortion 0.000

An example configuration where the noise evaluates to a constant value.

It seems all objects have a value of 0.5. To understand why this happens, let us look at the following plot of a 1D noise texture.



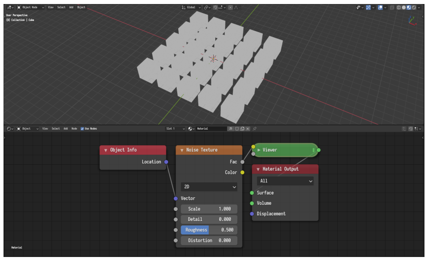
A plot of a 1D noise with zero details and zero distortion.

The horizontal line denotes a value of 0.5 and the vertical lines denotes whole numbers assuming a noise scale of 1. As can be seen, the noise always intersects the 0.5 line at whole numbers. Since the aforementioned objects were distributed on a grid and have whole number locations, they all evaluate to 0.5. Which explains the issue at hand.

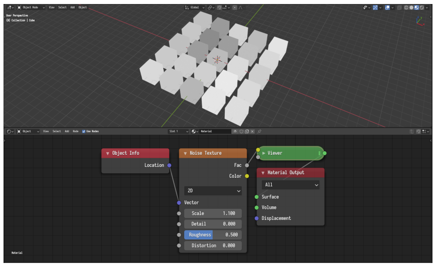
Generally, any discrete evaluation of noise at integer multiples of the reciprocal of the noise scale will always evaluate to 0.5. It also follows that evaluations closer to that will have values close to 0.5. In such cases, it is almost always preferred to use the White Noise Texture.

Regardless, one can mitigate this issue in a number of ways:

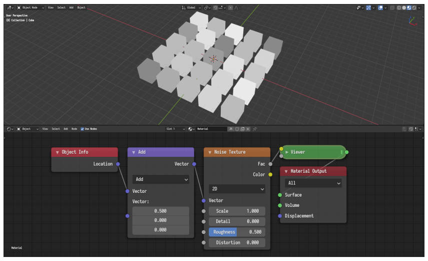
- Adjust the scale of the noise to avoid aligning the noise with the evaluation domain.
- Add an arbitrary offset to the texture coordinates to break the alignment with the evaluation domain.
- Evaluate the noise at a higher dimension and adjust the extra dimension until a satisfactory result is achieved.


Object Info
Location
Noise Texture
Fac
Color
Vector
Scale 1.000
Detail 0.000
Roughness 0.500
Distortion 0.000
Viewer
Material Output
All
Surface
Volume
Displacement

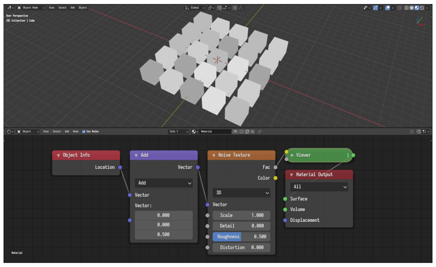
Constant value issue.


Object Info
Location
Noise Texture
Fac
Color
Vector
Scale 1.100
Detail 0.000
Roughness 0.500
Distortion 0.000
Viewer
Material Output
All
Surface
Volume
Displacement

Mitigating the issue by adjusting the scale.

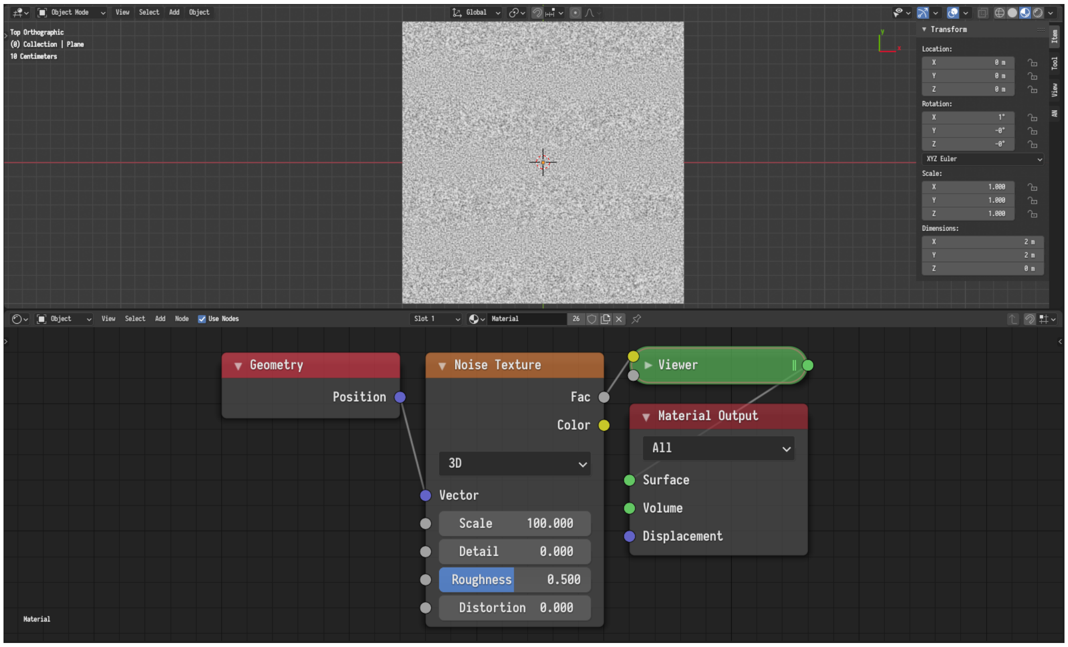

Object Info
Location
Add
Vector
Add
Vector
Vector:
0.500
0.000
0.000
Noise Texture
Fac
Color
Vector
Scale 1.000
Detail 0.000
Roughness 0.500
Distortion 0.000
Viewer
Material Output
All
Surface
Volume
Displacement

Mitigating the issue by adding an arbitrary offset.


Object Info
Location
Add
Vector
Add
Vector
Vector:
0.000
0.000
0.500
Noise Texture
Fac
Color
Vector
Scale 1.000
Detail 0.000
Roughness 0.500
Distortion 0.000
Viewer
Material Output
All
Surface
Volume
Displacement

Mitigating the issue by evaluating at a higher dimension.

Similarly, in other configurations, one might experience some banding patterns in the noise, where there are bands of high contrast areas followed by banding of low contrast areas. For instance, planar surfaces that are slightly tilted along one of the axis will have such a banding pattern.


Object Mode
View
Select
Add
Object
Global
Top Orthographic
(0) Collection | Plane
10 Centimeters
Item
Tool
View
Object
View
Select
Add
Node
Use Nodes
Slot 1
Material
Geometry
Position
Noise Texture
Fac
Color
Vector
Viewer
Material Output
Surface
Volume
Displacement
Material

Transform

Location:

X 0 m
Y 0 m
Z 0 m

Rotation:

X 1°
Y -0°
Z -0°
XYZ Euler

Scale:

X 1.000
Y 1.000
Z 1.000

Dimensions:

X 2 m
Y 2 m
Z 0 m

Noise Texture: 3D

Scale 100.000
Detail 0.000
Roughness 0.500
Distortion 0.000

Material Output: All

An example configuration where the noise have a banding pattern.

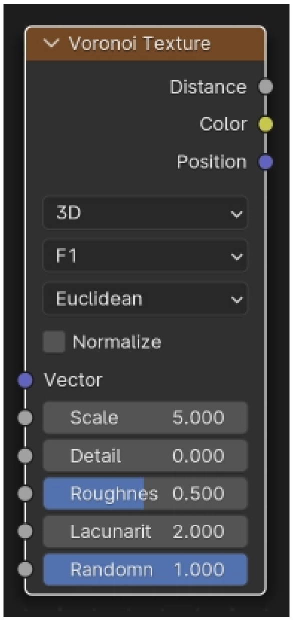
This happens because the slight tilt along one of the axis causes values along the perpendicular axis to change very slowly making the grid structure of the noise more apparent. The easiest way to mitigate this issue to rotate the coordinates by an arbitrary amount.

Voronoi Texture Node

Note

This node is ported from shader nodes. The manual and images are referencing the shader version of the node. This node accepts field inputs and outputs. When not connected the Vector input has an implicit `position` attribute value.

The *Voronoi Texture* node evaluates a [Worley Noise](https://en.wikipedia.org/wiki/Worley_noise) at the input texture coordinates.



3D
F1
Euclidean
☐ Normalize
Scale 5.000
Detail 0.000
Roughnes 0.500
Lacunarit 2.000
Randomn 1.000

Inputs

The inputs are dynamic, they become available if needed depending on the node properties.

Vector
: Texture coordinate to evaluate the noise at; defaults to *Generated* texture coordinates if the socket is left unconnected.

W
: Texture coordinate to evaluate the noise at.

Scale
: Scale of the noise.

Detail
: Number of noise octaves. The fractional part of the input is multiplied by the magnitude of the highest octave. Higher number of octaves corresponds to a higher evaluation time.

Roughness

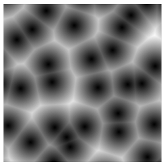
Blend between a smoother noise pattern, and rougher with sharper peaks.

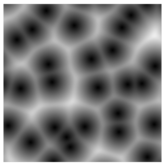
Lacunarity

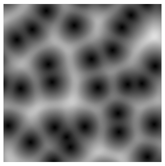
The difference between the scale of each two consecutive octaves. Larger values corresponds to larger scale for higher octaves.

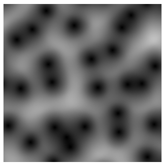
Smoothness

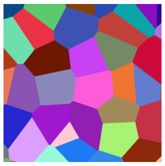
The smoothness of the noise.

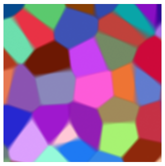

Smoothness: 0.0.

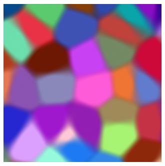

Smoothness: 0.25.

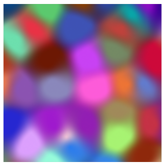

Smoothness: 0.5.


Smoothness: 1.0.


Smoothness: 0.0.

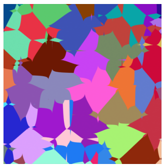

Smoothness: 0.25.

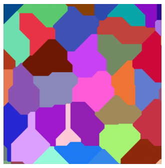

Smoothness: 0.5.

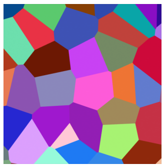

Smoothness: 1.0.

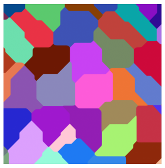
Exponent

Exponent of the Minkowski distance metric.


Exponent: 0.5.

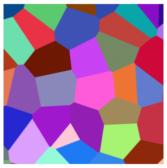

Exponent: 1.0.

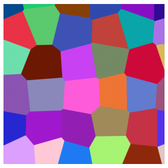

Exponent: 2.0.

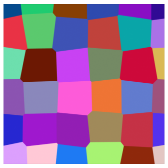

Exponent: 32.0.

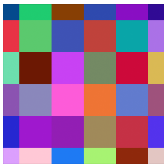
Randomness

The randomness of the noise.









Randomness: 1.0.

Randomness: 0.5.

Randomness: 0.25.

Randomness: 0.0.

Properties

Dimensions

The dimensions of the space to evaluate the noise in.

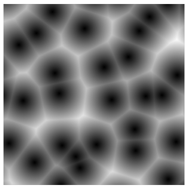
1D::	Evaluate the noise in 1D space at the input W.
2D::	Evaluate the noise in 2D space at the input Vector. The Z component is ignored.
3D::	Evaluate the noise in 3D space at the input Vector.
4D::	Evaluate the noise in 4D space at the input Vector and the input W as the fourth dimension.

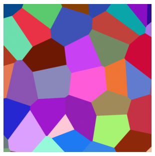
Higher dimensions corresponds to higher render time, so lower dimensions should be used unless higher dimensions are necessary.

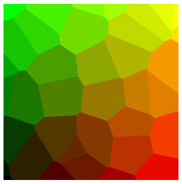
Feature

The Voronoi feature that the node will compute.

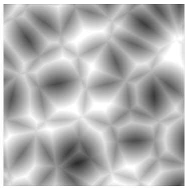
F1:: The distance to the closest feature point as well as its position and color.

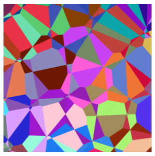

Distance.

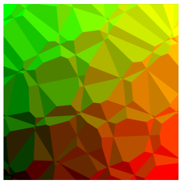

Color.


Position.

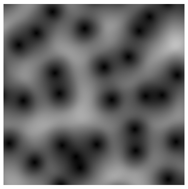
F2:: The distance to the second closest feature point as well as its position and color.

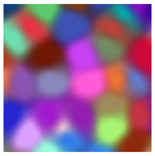

Distance.

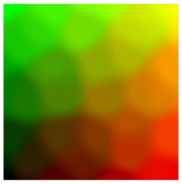

Color.


Position.

Smooth F1:: A smooth version of F1.

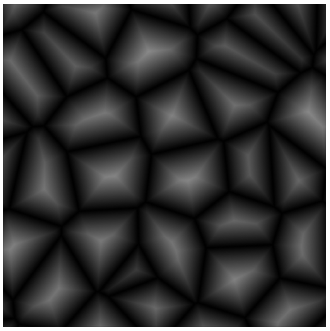



Color.

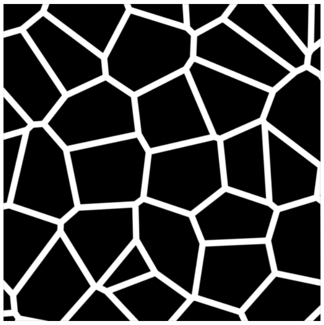


Distance. Position.

Distance to Edge:: The distance to the edges of the Voronoi cells.

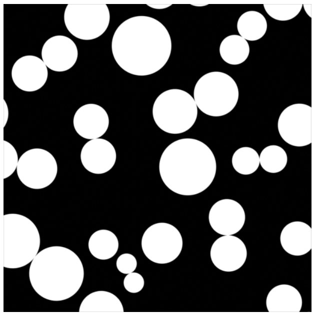


Distance.

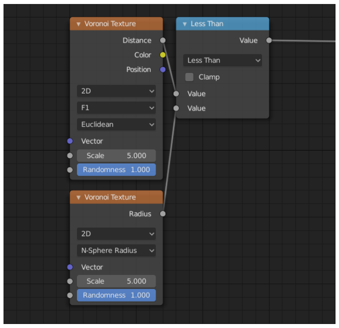


Distance smaller than 0.05.

N-Sphere Radius:: The radius of the n-sphere inscribed in the Voronoi cells. In other words, it is half the distance between the closest feature point and the feature point closest to it.



The n-sphere radius can be used to create tightly packed n-spheres.

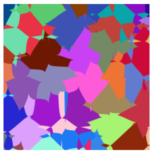

Voronoi Texture
Distance
Color
Position
2D
F1
Euclidean
Vector
Scale 5.000
Randomness 1.000
Less Than
Value
Less Than
Clamp
Value
Value
Voronoi Texture
Radius
2D
N-Sphere Radius
Vector
Scale 5.000
Randomness 1.000

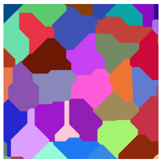
Node tree for the shader to the left.

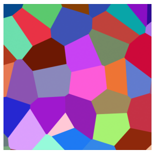
Distance Metric

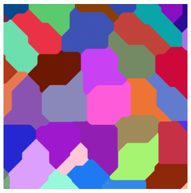
The distance metric used to compute the texture.

Euclidean::	Use the Euclidean distance metric.
Manhattan::	Use the Manhattan distance metric.
Chebychev::	Use the Chebychev distance metric.
Minkowski::	Use the Minkowski distance metric. The Minkowski distance is a generalization of the aforementioned metrics with an *Exponent* as a parameter. Minkowski with an exponent of one is equivalent to the *Manhattan* distance metric. Minkowski with an exponent of two is equivalent to the *Euclidean* distance metric. Minkowski with an infinite exponent is equivalent to the *Chebychev* distance metric.


Minkowski Exponent: 0.5 (Minkowski 1/2).


Minkowski Exponent: 1.0 (Manhattan).


Minkowski Exponent: 2.0 (Euclidean).


Minkowski Exponent: 32.0 (approximation of Chebychev).

Normalize

If enabled, ensures that the output values stay in the range 0.0 to 1.0. In rare cases, the output value may be outside that range when *Feature* is *F2*.

Outputs

Distance

Distance.

Color

Cell color. The color is arbitrary.

Position

Position of feature point.

W

Position of feature point.

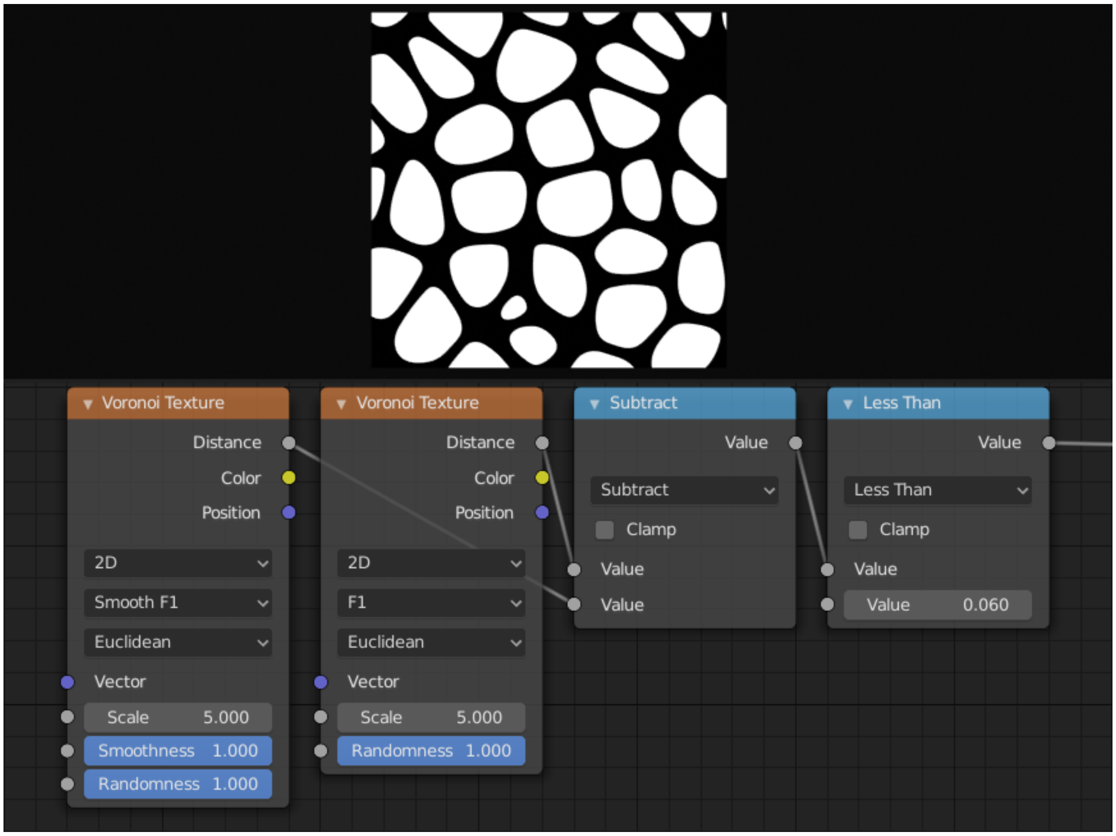
Radius

N-Sphere radius.

Notes

In some configurations of the node, especially for low values of *Randomness*, rendering artifacts may occur. This happens due to the same reasons described in the Notes section in the White Noise Texture page and can be fixed in a similar manner as described there.

Examples


Voronoi Texture
Distance
Color
Position
Vector
Voronoi Texture
Distance
Color
Position
Vector
Subtract
Value
Value
Value
Less Than
Value
Value

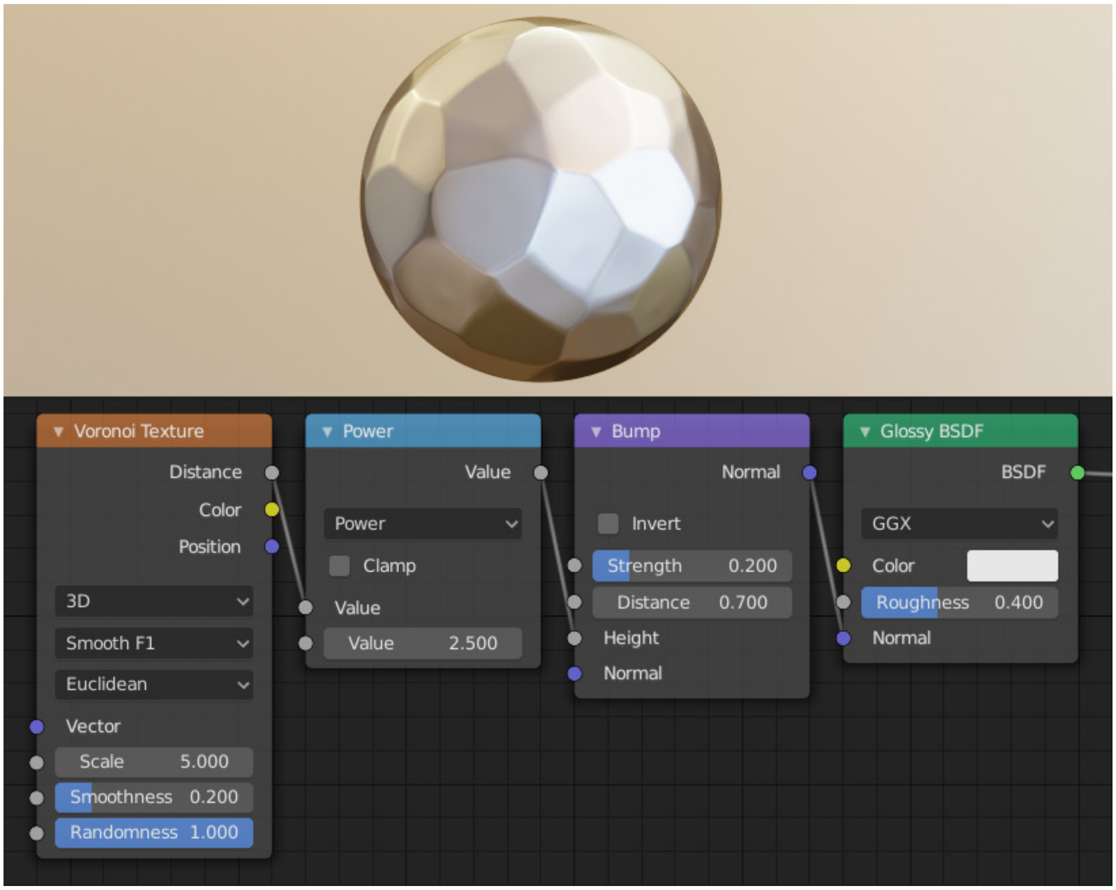
2D
Smooth F1
Euclidean
Scale 5.000
Smoothness 1.000
Randomness 1.000

2D
F1
Euclidean
Scale 5.000
Randomness 1.000

Subtract
☐ Clamp

Less Than
☐ Clamp
Value 0.060

The difference between *F1* and *Smooth F1* can be used to create beveled Voronoi cells.


Voronoi Texture
Distance
Color
Position
3D
Smooth F1
Euclidean
Vector
Scale 5.000
Smoothness 0.200
Randomness 1.000

Power
Value
Power
☐ Clamp
Value
Value 2.500

Bump
Normal
☐ Invert
Strength 0.200
Distance 0.700
Height
Normal

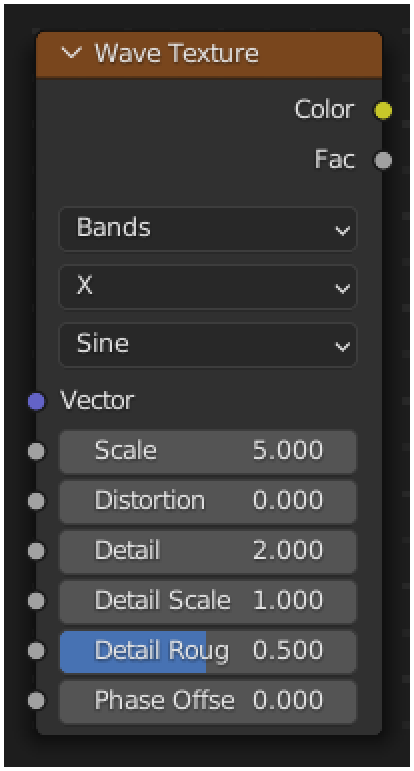
Glossy BSDF
BSDF
GGX
Color
Roughness 0.400
Normal

Creating a hammered metal shader using the *Voronoi Texture* node.

Wave Texture Node

Note

This node is ported from shader nodes. The manual and images are referencing the shader version of the node. This node accepts field inputs and outputs. When not connected the Vector input has an implicit `position` attribute value.



Bands
X
Sine
Scale 5.000
Distortion 0.000
Detail 2.000
Detail Scale 1.000
Detail Roug 0.500
Phase Offse 0.000

The *Wave Texture* node adds procedural bands or rings with noise distortion.

Hint

In general, textures can be distorted by mixing their texture coordinates with another texture. The distortion built into the *Wave Texture Node* uses the *Color* output of the Noise Texture Node.

To replicate this, center its value range around zero, multiply it by a factor proportional to *Distortion*/*Scale* and add the result onto the texture coordinates. *Detail*, *Detail Scale* and *Roughness* of the *Wave Texture Node* correspond to the inputs on the Noise Texture Node.

Inputs

Vector

Texture coordinate to sample texture at; defaults to Generated texture coordinates if the socket is left unconnected.

Scale

Overall texture scale.

Distortion

Amount of distortion of the wave.

Hint

In general, textures can be distorted by mixing their texture coordinates with another texture. The distortion built into the *Wave Texture Node* uses the *Color* output of the Noise Texture Node.

To replicate this, center its value range around zero, multiply it by a factor proportional to *Distortion*/*Scale* and add the result onto the texture coordinates. *Detail*, *Detail Scale*, and *Roughness* of the *Wave Texture Node* correspond to the inputs on the Noise Texture Node.

Detail

Amount of distortion noise detail.

Detail Scale

Scale of distortion noise.

Roughness

Blend between a smoother noise pattern, and rougher with sharper peaks.

Phase Offset

Position of the wave along the *Bands Direction*. This can be used as an input for more control over the distortion.

Properties

Type

Bands or *Rings* shaped waves.

Bands/Rings Direction

The axis the bands or rings propagate from i.e. which axis they are perpendicular to. When using *Bands* a *Diagonal* axis is an option and when using *Rings* the rings can propagate outwards from a single point by using *Spherical* direction.

Wave Profile

Controls the look of the wave type.

Saw::	Uses a sawtooth profile.
Sine::	Uses the standard sine profile.

Outputs

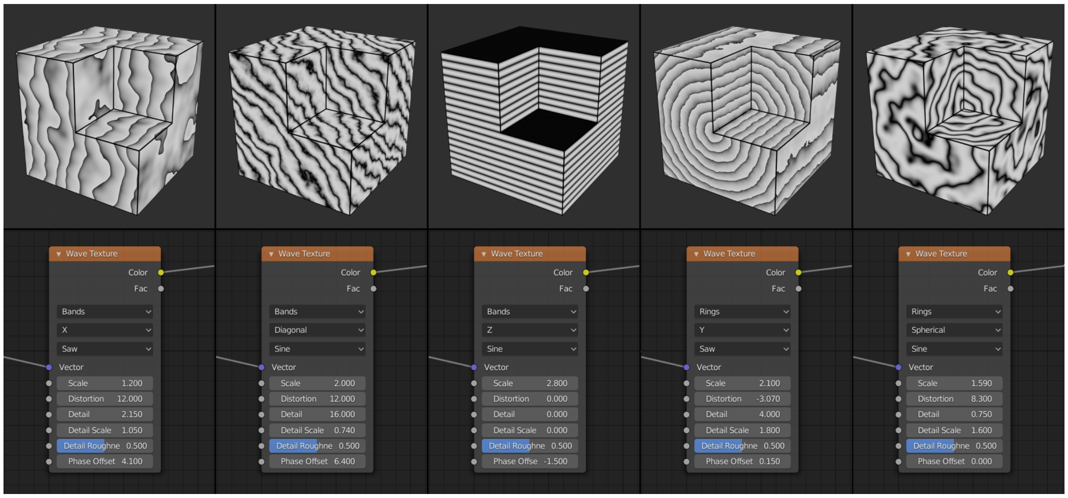
Color

Texture color output.

Factor

Texture intensity output.

Examples



	Bands	Bands	Bands	Rings	Rings
	X	Diagonal	Z	Y	Spherical
	Saw	Sine	Sine	Saw	Sine
Scale	1.200	2.000	2.800	2.100	1.590
Distortion	12.000	12.000	0.000	-3.070	8.300
Detail	2.150	16.000	0.000	4.000	0.750
Detail Scale	1.050	0.740	0.000	1.800	1.600
Detail Roughne	0.500	0.500	0.500	0.500	0.500
Phase Offset	4.100	6.400	-1.500	0.150	0.000

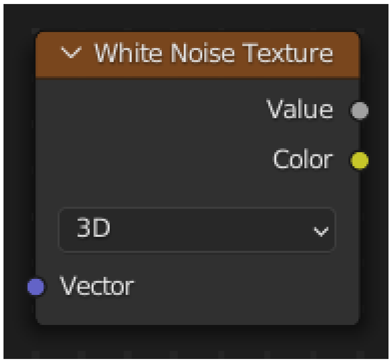
Wave Texture.

White Noise Texture Node

Note

This node is ported from shader nodes. The manual and images are referencing the shader version of the node. This node accepts field inputs and outputs. When not connected the Vector input has an implicit `position` attribute value.

The *White Noise Texture* node returns a random number based on an input Seed. The seed can be a number, a 2D vector, a 3D vector, or a 4D vector; depending on the *Dimensions* property. The output number ranges between zero and one.



Inputs

The inputs are dynamic, they become available if needed depending on the node properties.

Vector
    Vector used as seed in 2D, 3D, and 4D dimensions.

W
    Value used as seed in 1D and 4D dimensions.

Properties

Dimensions

The dimensions of the space to evaluate the noise in.

1D::	The *W* input is used as seed.
2D::	The X and Y components of the *Vector* input are used as seed.
3D::	The *Vector* input is used as seed.
4D::	Both the *Vector* input and the *W* input are used as seed.

Outputs

Value

Output random value.

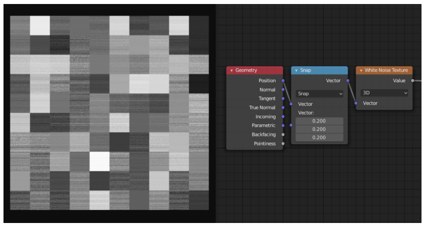
Color

Output random color.

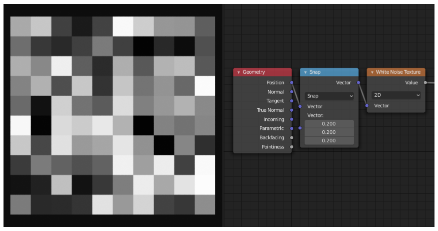
Notes

The slightest difference in seed values would result in completely different outputs. Consequently, bad precision may have significant impact on the output. Usually, we can mitigate this issue by:

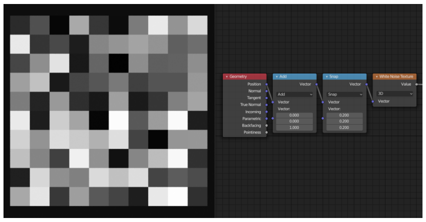
- Eliminating the problematic seed value. If the problematic seed value is constant, it should be eliminated by choosing a lower dimension or multiplying it by zero.
- Adding an arbitrary value to the seed. The issue might only happen at certain boundaries, like unit boundaries, so simply adding an arbitrary value might solve the issue.
- Taking the absolute value of the seed. In computing, zero may be positive or negative, so taking the absolute values unifies the zero into a single value.



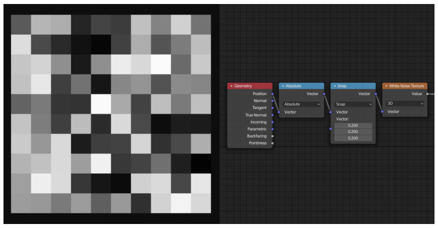
Precision issue due to signed zeros on the Z axis.



Mitigating the issue by eliminating the Z axis.

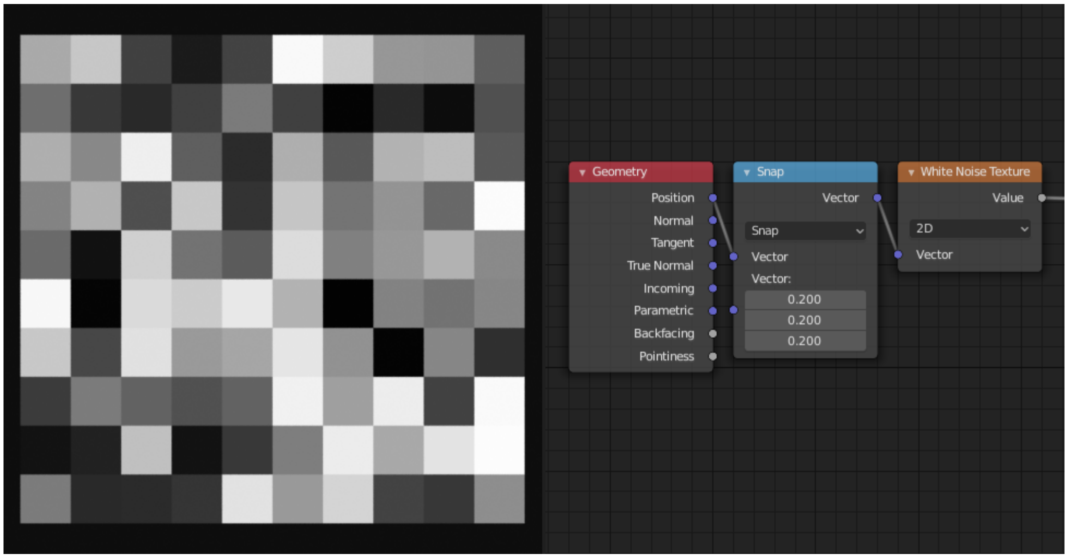


Mitigating the issue by adding an arbitrary value.



Mitigating the issue by taking the absolute value.

Examples


Geometry
Position
Normal
Tangent
True Normal
Incoming
Parametric
Backfacing
Pointiness
Snap
Vector
Snap
Vector
Vector:
0.200
0.200
0.200
White Noise Texture
Value
2D
Vector

Generating cell noise using the *Snap* vector operation and the *White Noise* node.

Utilities Nodes

General purpose nodes for modifying data.

- Color
    - Blackbody Node
    - Color Ramp Node
    - Combine Color Node
    - Mix Color Node
    - RGB Curves Node
    - Separate Color Node
- Text
    - Join Strings Node
    - Replace String Node
    - Slice String Node
    - Special Characters Node
    - String Length Node
    - String to Curves Node
    - Value to String Node
- Vector
    - Vector Curves Node
    - Vector Math Node
    - Vector Rotate Node
    - Combine XYZ Node
    - Mix Vector Node
    - Separate XYZ Node

- Field
    - Accumulate Field Node
    - Evaluate at Index Node
    - Evaluate on Domain Node
- Math
    - Boolean Math Node
    - Clamp Node

  - Compare Node
  - Float Curve
  - Float To Integer Node
  - Hash Value Node
  - Integer Math Node
  - Map Range Node
  - Math Node
  - Mix Node
- Matrix
  - Combine Matrix Node
  - Combine Transform Node
  - Invert Matrix Node
  - Matrix Determinant Node
  - Multiply Matrices Node
  - Project Point Node
  - Separate Matrix Node
  - Separate Transform Node
  - Transform Direction Node
  - Transform Point Node
  - Transpose Matrix Node
- Rotation
  - Align Rotation to Vector Node
  - Axes to Rotation Node
  - Axis Angle to Rotation Node
  - Euler to Rotation Node
  - Invert Rotation Node
  - Rotate Rotation Node
  - Rotate Vector Node
  - Rotation to Euler Node
  - Rotation to Quaternion Node
  - Quaternion to Rotation Node
- Deprecated
  - Align Euler to Vector Node
  - Rotate Euler Node

- For Each Geometry Element Zone
- Index Switch Node

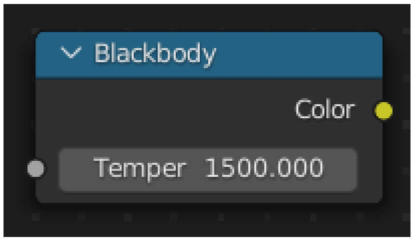
- Menu Switch Node
- Random Value Node
- Repeat Zone
- Switch Node

Color Utility Nodes

Nodes for modifying color data passed through color sockets.

- Blackbody Node
- Color Ramp Node
- Combine Color Node
- Mix Color Node
- RGB Curves Node
- Separate Color Node

Blackbody Node


Blackbody
Color
Temper 1500.000

The *Blackbody* node converts a blackbody temperature to RGB value. This can be useful for materials that emit light at natural occurring frequencies.

Inputs

Temperature
  The temperature in Kelvin.

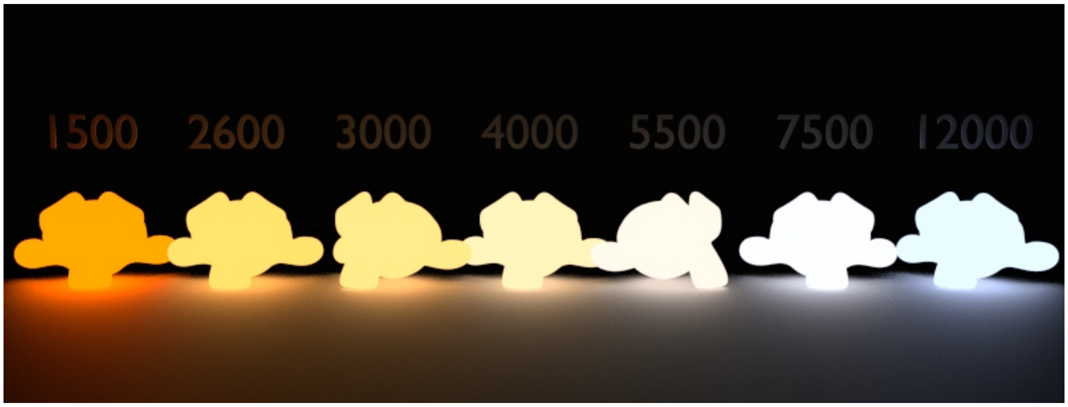
Properties

This node has no properties.

Outputs

Color
  RGB color output.

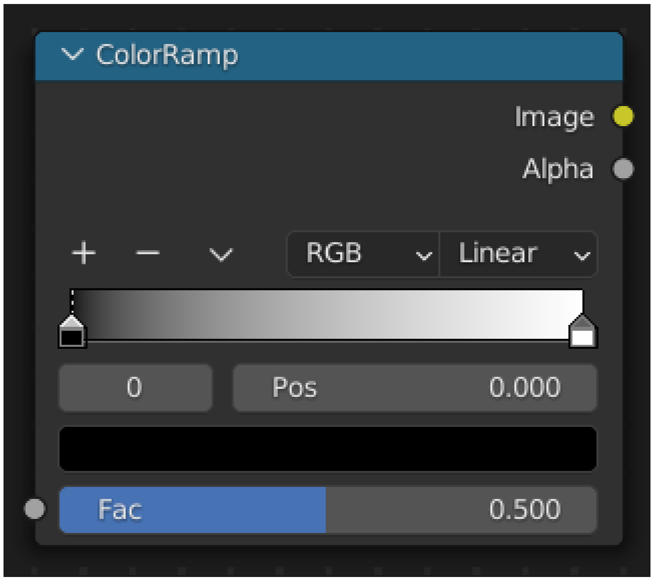
Examples


1500
2600
3000
4000
5500
7500
12000

Example of the color ranges of the Blackbody node.

Color Ramp Node

The Color Ramp Node is used for mapping values to colors using a gradient.



RGB
Linear
0
Pos 0.000
Fac 0.500

Inputs

Factor

The value to map. 0.0 results in the leftmost color, while 1.0 results in the rightmost.

Properties

Color Ramp

See Color Ramp Widget.

Outputs

Image/Color

Standard color output.

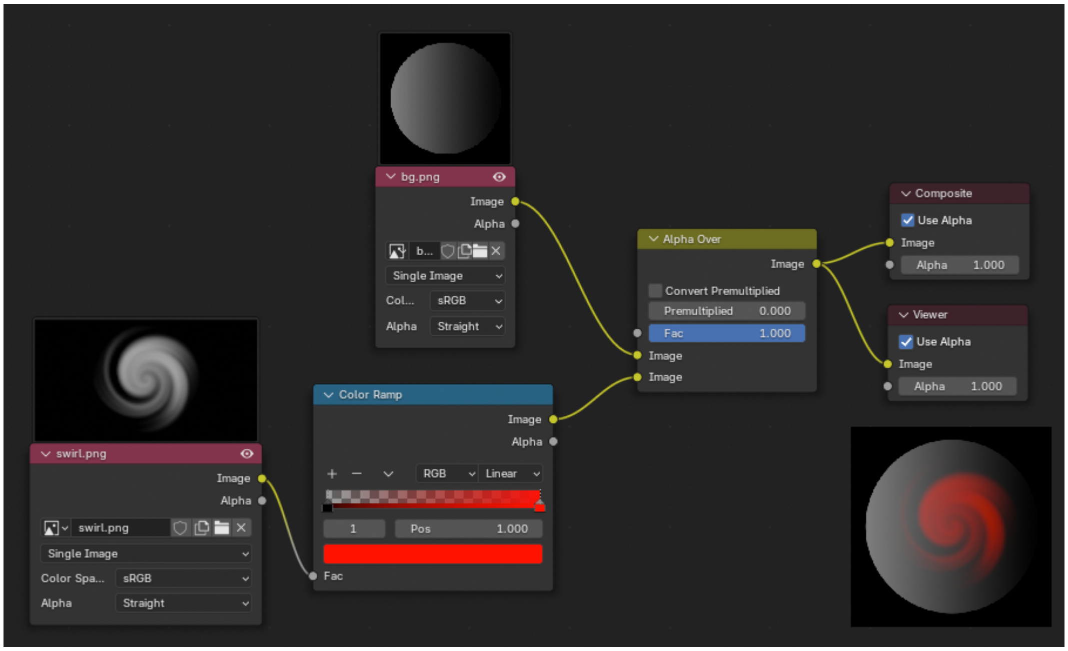
Alpha

Standard alpha output.

Examples

Creating an Alpha Mask

An often overlooked use case of the Color Ramp is to turn a black-and-white image into a colored image with transparency.

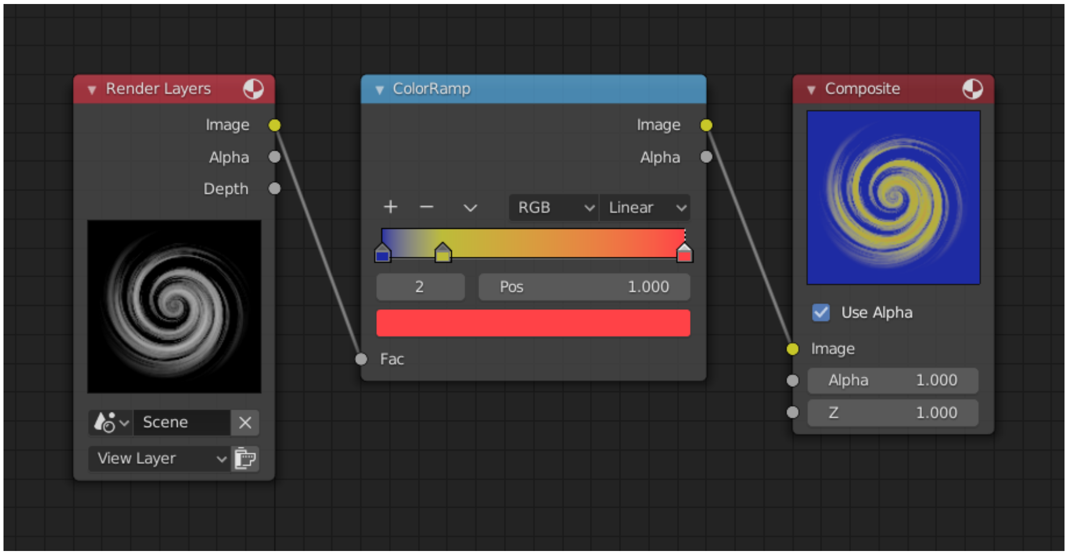

bg.png
Image
Alpha
Single Image
Col... sRGB
Alpha Straight
swirl.png
Image
Alpha
swirl.png
Single Image
Color Spa... sRGB
Alpha Straight
Color Ramp
Image
Alpha
RGB
Linear
1
Pos 1.000
Fac
Alpha Over
Image
Convert Premultiplied
Premultiplied 0.000
Fac 1.000
Image
Image
Composite
Use Alpha
Image
Alpha 1.000
Viewer
Use Alpha
Image
Alpha 1.000

In the example above, a black-and-white swirl image, which is lacking an alpha channel, is fed into the Color Ramp node as a *Factor*.

The Color Ramp node is set to a purely transparent color on the left end of the gradient, and a fully red color on the right. As you can see in the Viewer node, the Color Ramp node outputs an image that is transparent where the input is black, and opaque where the input is white.

Colorizing an Image

In this example, multiple colors are added to the color gradient, converting a black-and-white image into a flaming swirl.


Render Layers
Image
Alpha
Depth
Scene
View Layer
ColorRamp
Image
Alpha
Fac
Composite
Image

RGB Linear

2 Pos 1.000

☑ Use Alpha

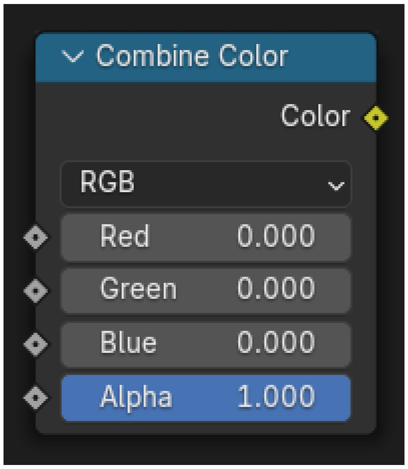
Alpha 1.000

Z 1.000

The shades of gray in the input image are mapped to three colors: blue, yellow, and red, all fully opaque. Where the image is black, the Color Ramp substitutes blue (the first color stop). Where it is some shade of gray, the Color Ramp outputs a corresponding color from the gradient (bluish, yellow, to reddish). Where the image is fully white, the Color Ramp outputs red.

Combine Color Node

Combines four grayscale channels into one color image, based on a particular Color Model.



RGB
Red 0.000
Green 0.000
Blue 0.000
Alpha 1.000

Inputs

The inputs of this node depend on the Mode property (see below).

Alpha
: The opacity of the output color.

Properties

Mode
: The color model to use.

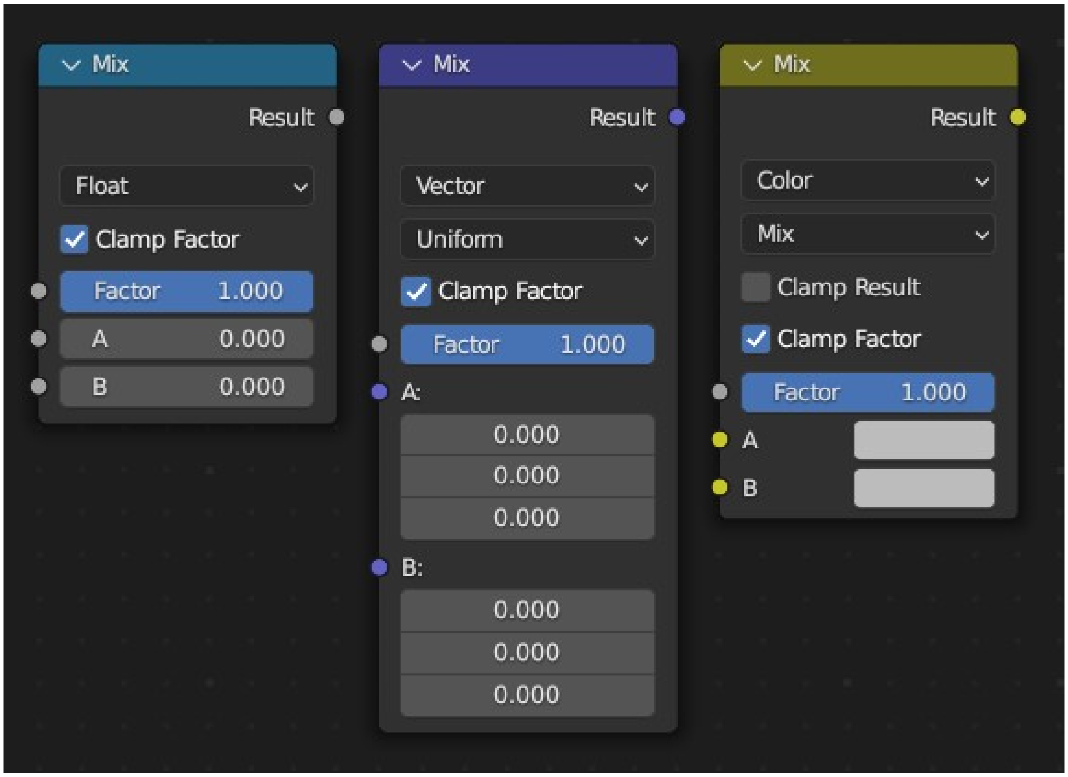
RGB::	Red, Green, Blue.
HSV::	Hue, Saturation, Value.
HSL::	Hue, Saturation, Lightness.

Output

Color
: Standard color output.

Mix Color Node

The *Mix Node* mixes values, colors and vectors inputs using a factor to control the amount of interpolation. The *Color* mode has additional blending modes.


Mix
Result
Float
Clamp Factor
Factor 1.000
A 0.000
B 0.000
Mix
Result
Vector
Uniform
Clamp Factor
Factor 1.000
A:
0.000
0.000
0.000
B:
0.000
0.000
0.000
Mix
Result
Color
Mix
Clamp Result
Clamp Factor
Factor 1.000
A
B

Inputs

Factor
: Controls the amount of mixing between the A and B inputs.

A/B
: The two inputs that are mixed together.

Properties

Data Type

The data type that is used for mixing. The node supports float, vector, color, and rotation data types.

Factor Mode (Vector only)

The factor mode can be set to *Uniform* and *Non-Uniform*. In uniform mode, a single float controls the factor. In non-uniform mode, a vector controls the factor for each XYZ channel separately.

Mix (Color only)

The Blend modes can be selected in the select menu. See Color Blend Modes for details on each blending mode.

Add, Subtract, Multiply, Screen, Divide, Difference, Darken, Lighten, Overlay, Color Dodge, Color Burn, Hue, Saturation, Value, Color, Soft Light, Linear Light

Clamp Factor

Limit the factor value between 0.0 and 1.0. If this option is unchecked then the node operates using *Extrapolation*.

Clamp Result (Color only)

Limit the Result to the range between 0.0 and 1.0.

Outputs

Result

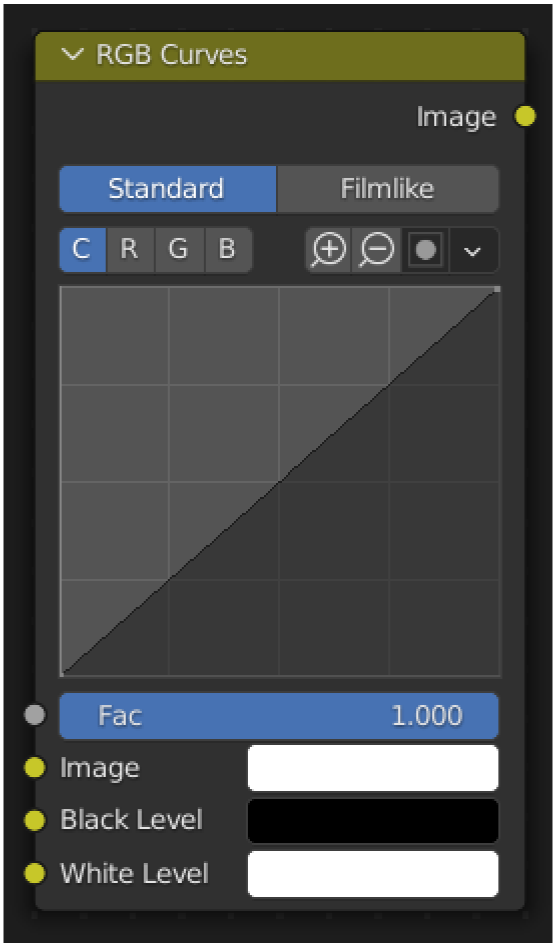
Output the result of the mix using the data type selected.

Examples

See the Color > Mix page for additional examples: Mix Color Node

RGB Curves Node

The *RGB Curves Node* performs level adjustments on each color channel.


RGB Curves
Image
Standard
Filmlike
C
R
G
B
Image
Black Level
White Level

Fac 1.000

Inputs

Factor
: Controls the amount of influence the node exerts on the image.

Image/Color
: Standard color input.

Black Level *Compositor Only*
: Defines the input color that should be mapped to black.

White Level *Compositor Only*

Defines the input color that should be mapped to white.

Tip

To define the black and white levels, use the eyedropper to select a color sample of a displayed image.

Properties

Tone *Compositor Only*

Standard::	The Combined curve is applied to each channel individually, which may result in a change of hue.
Filmlike::	Keeps the hue constant.

Channel

The curve to show.

C::	Combined
R::	Red
G::	Green
B::	Blue

Curve

A Bézier curve that maps each input level (X axis) to an output level (Y axis). For the curve controls, see Curve widget.

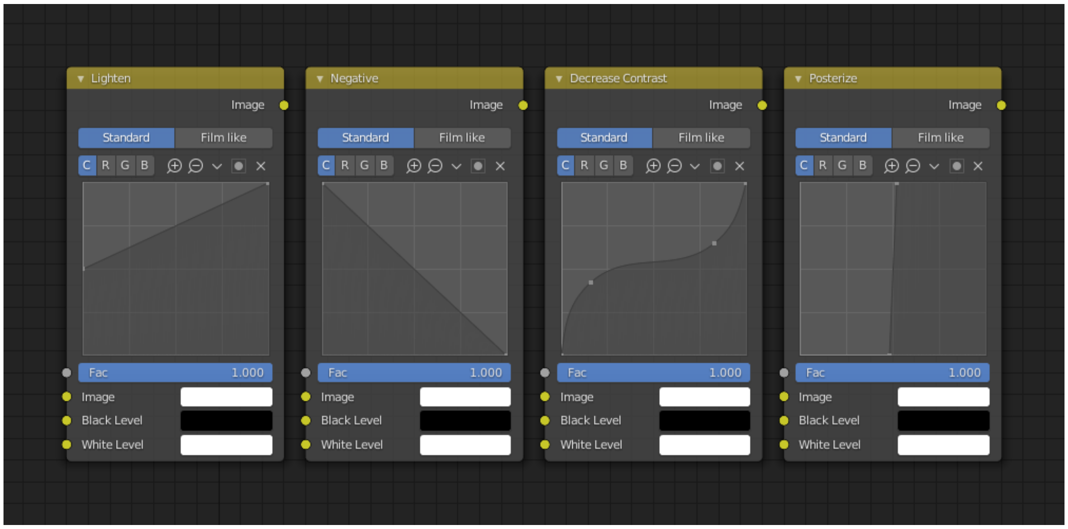
Outputs

Image/Color

Standard color output.

Examples

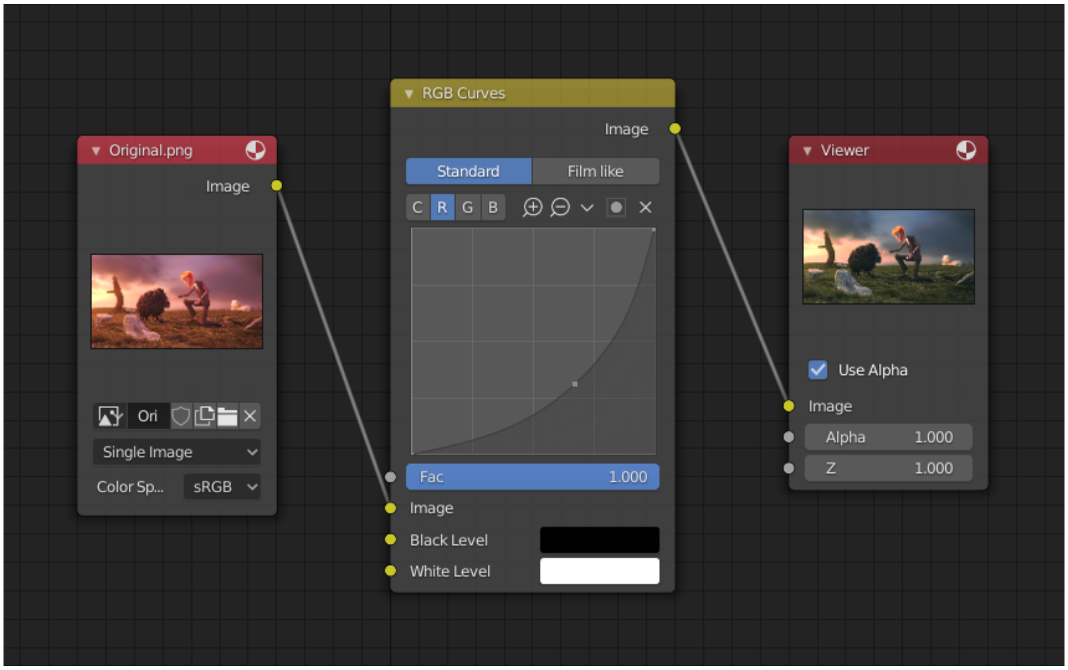
Below are some common curves you can use to achieve desired effects.


Lighten
Negative
Decrease Contrast
Posterize
Image
Standard
Film like
C R G B
Image
Black Level
White Level

Fac 1.000

From left to right: 1. Lighten shadows 2. Negative 3. Decrease contrast 4. Posterize.

Color Correction using Curves


Original.png
Image
Ori
RGB Curves
Image
Standard
Film like
C R G B
Image
Black Level
White Level
Viewer
Use Alpha
Image

Single Image

Color Sp... sRGB

Fac 1.000

☑ Use Alpha

Alpha 1.000

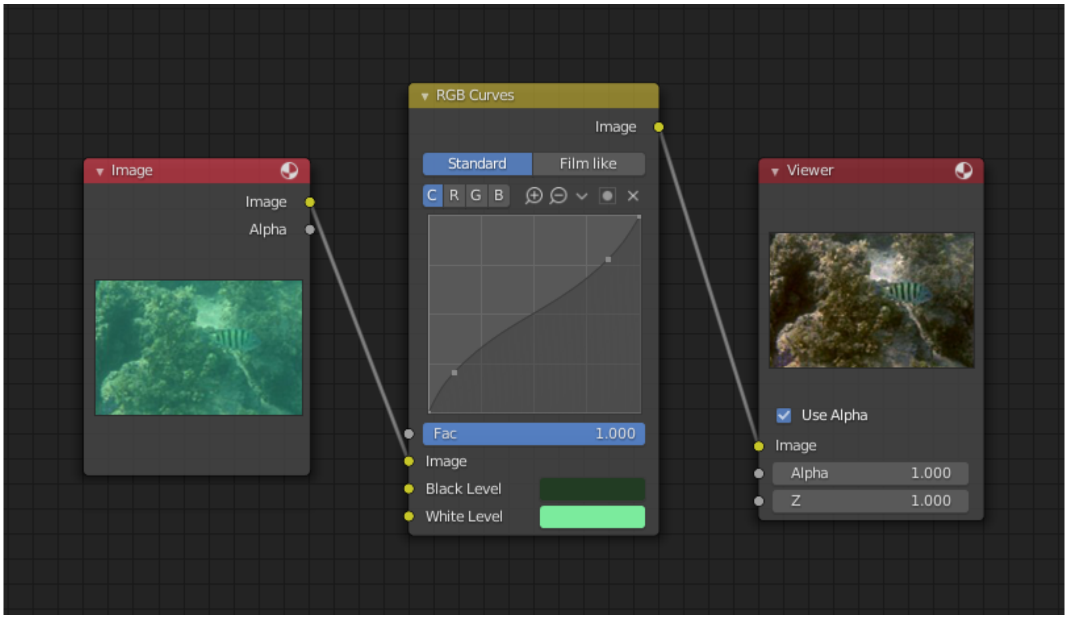
Z 1.000

Color correction with curves.

In this example, the image has too much red in it, so we run it through an *RGB Curves* node and reduce the Red channel.

The documentation for the Mix Node has an additional example about fixing overexposure.

Color Correction using Black/White Levels



Fac 1.000

☑ Use Alpha

Alpha 1.000

Z 1.000

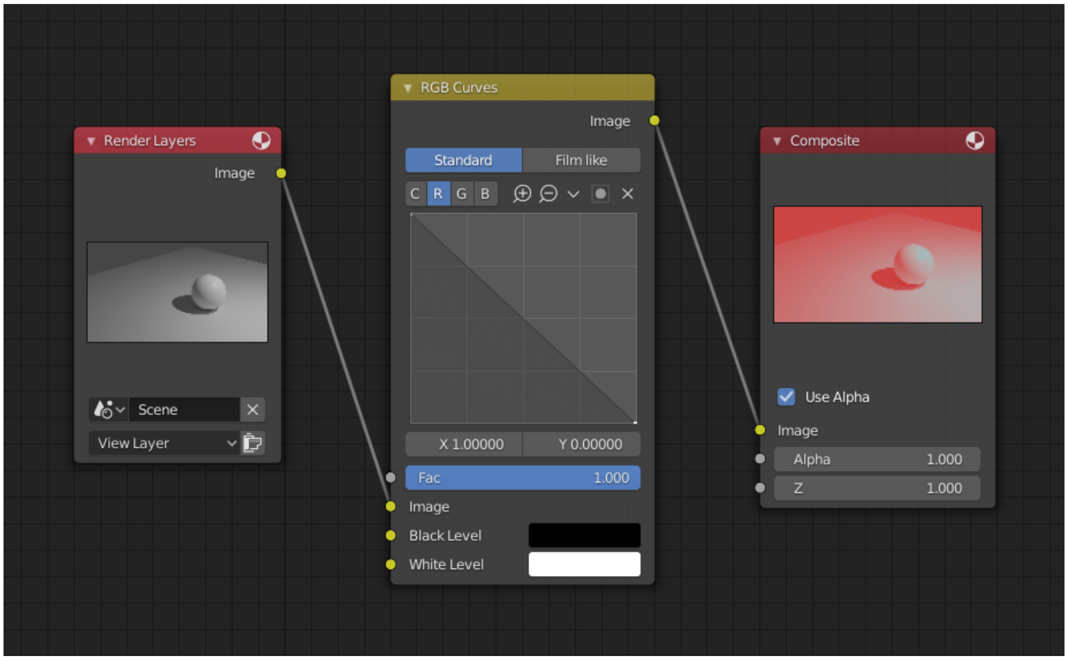
Color correction with Black/White Levels.

Manually adjusting the RGB curves for color correction can be difficult. Another option for color correction is to use the Black and White Levels instead, which really might be their main purpose.

In this example, the White Level is set to the color of a bright spot of the sand in the background, and the Black Level to the color in the center of the fish's eye. To do this efficiently it is best to bring up the Image Editor showing the original input image. You can then use the levels' color picker to easily choose the appropriate colors from the input image, zooming into pixel level if necessary. The result can be fine-tuned with the R, G, and B curves like in the previous example.

The curve for C is used to compensate for the increased contrast that is a side effect of setting Black and White Levels.

Effects

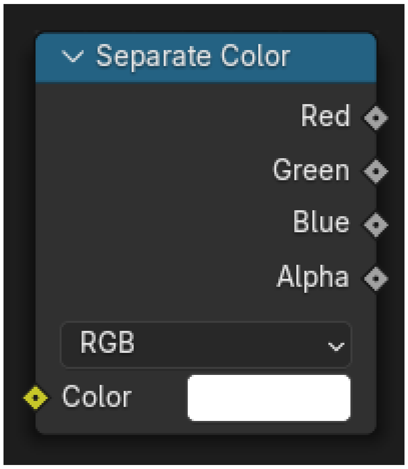

Render Layers
Image
Scene
View Layer
RGB Curves
Image
Standard
Film like
C
R
G
B
Image
Black Level
White Level
Composite
Image

X 1.00000
Y 0.00000
Fac 1.000
☑ Use Alpha
Alpha 1.000
Z 1.000

Changing colors by inverting the red channel.

Separate Color Node

Splits an image into its channels, based on a particular Color Model.


Separate Color
Red
Green
Blue
Alpha
RGB
Color

Inputs

Color
: Standard color input.

Properties

Mode
: The color model to output.

RGB::	Red, Green, Blue.
HSV::	Hue, Saturation, Value.
HSL::	Hue, Saturation, Lightness.

Outputs

The outputs of this node depend on the Mode property (see above).

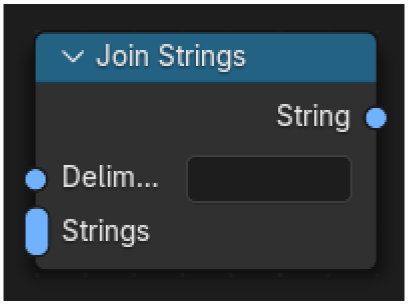
Alpha
: The opacity value.

Text Utility Nodes

Nodes to manipulate strings.

- Join Strings Node
- Replace String Node
- Slice String Node
- Special Characters Node
- String Length Node
- String to Curves Node
- Value to String Node

Join Strings Node



The *Join Strings* node combines any number of input strings into the output string. The order of the result depends on the vertical ordering of the inputs in the multi-input socket.

Tip

This node can be used to create a multi-line string for the String to Curves Node, when combined with the line break output from the Special Characters Node.

Inputs

Delimiter
String value to place between each concatenated string.

Strings
Multiple string values to be combined in connection order.

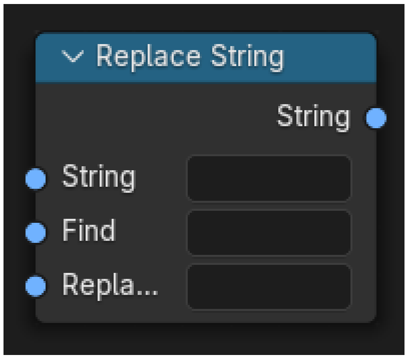
Properties

This node has no properties.

Outputs

String
  String result of the concatenation.

Replace String Node


Replace String
String
String
Find
Repla...

The *Replace String* node replaces a string segment with another.

Inputs

String
: Standard string input.

Find
: The substring to find in *String* to be replaced.

Replace
: A string segment which replaces occurrences of the *Find* substring.

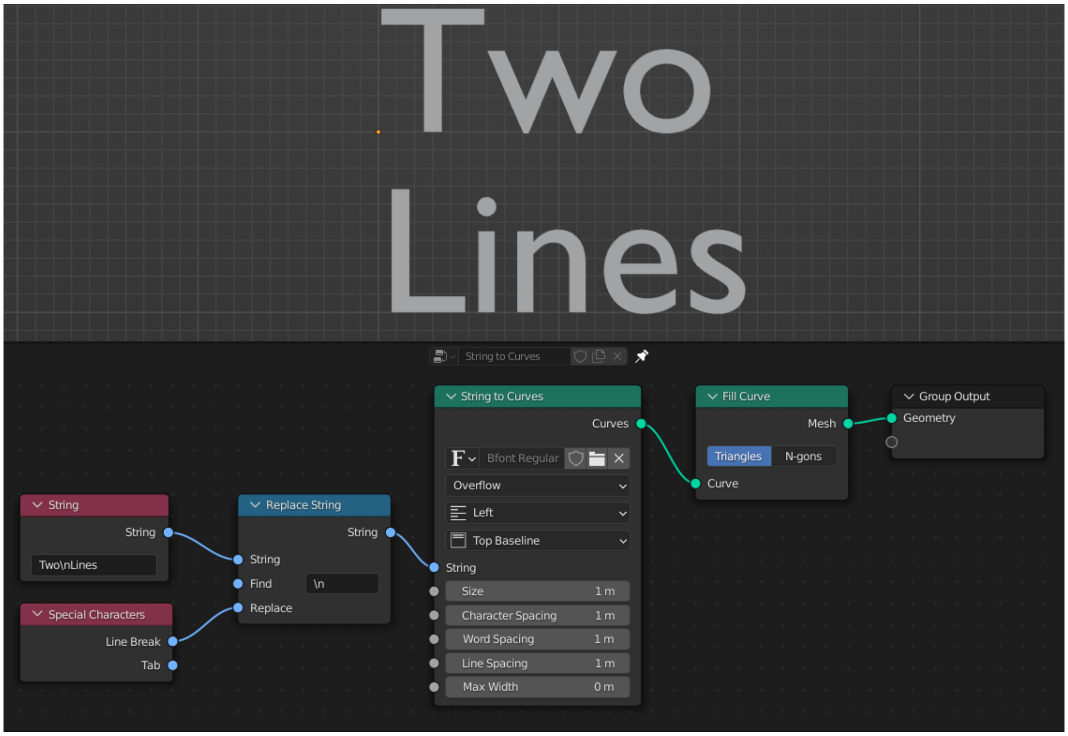
Properties

This node has no properties.

Outputs

String
: Standard string output.

Examples


Two
Lines
String to Curves
String
String
Special Characters
Line Break
Tab
Replace String
String
String
Find
Replace
String to Curves
Curves
Bfont Regular
Overflow
Left
Top Baseline
String
Fill Curve
Mesh
Triangles
N-gons
Curve
Group Output
Geometry

String: Two\nLines

Find: \n

Size 1 m

Character Spacing 1 m

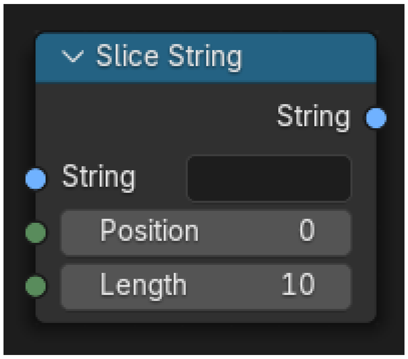
Word Spacing 1 m

Line Spacing 1 m

Max Width 0 m

Using the node to add the newline character to a string.

Slice String Node



Position 0

Length 10

The *Slice String* node extracts a string segment from a larger string.

Inputs

String
String value to be sliced.

Position
Integer value used to determine the starting point of the new string within the input string. The first letter of the string is at index 0.

Length
Integer value used to determine how many characters are extracted from the input string.

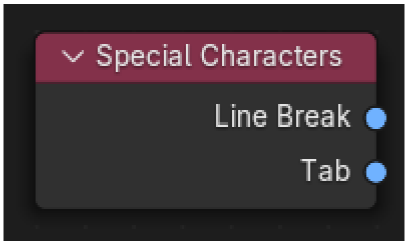
Properties

This node has no properties.

Outputs

String
String value of the extracted substring.

Special Characters Node



The *Special Characters* node is used to output string characters that can't be typed directly with the keyboard.

Tip

This node can be used to create a multi-line string for the String to Curves Node, when combined with the Join Strings Node or the Replace String Node.

Inputs

This node has no inputs.

Properties

This node has no properties.

Outputs

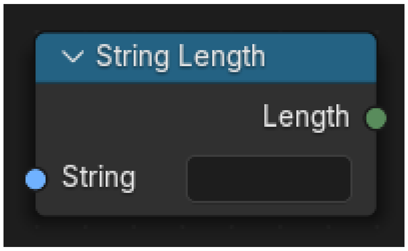
Line Break

A new line character (escape character `\n`).

Tab

A tab character used to add an indentation in the output.

String Length Node


String Length
Length
String

The *String Length* node outputs the number of characters in the input string.

Inputs

String
: String value to be evaluated.

Properties

This node has no properties.

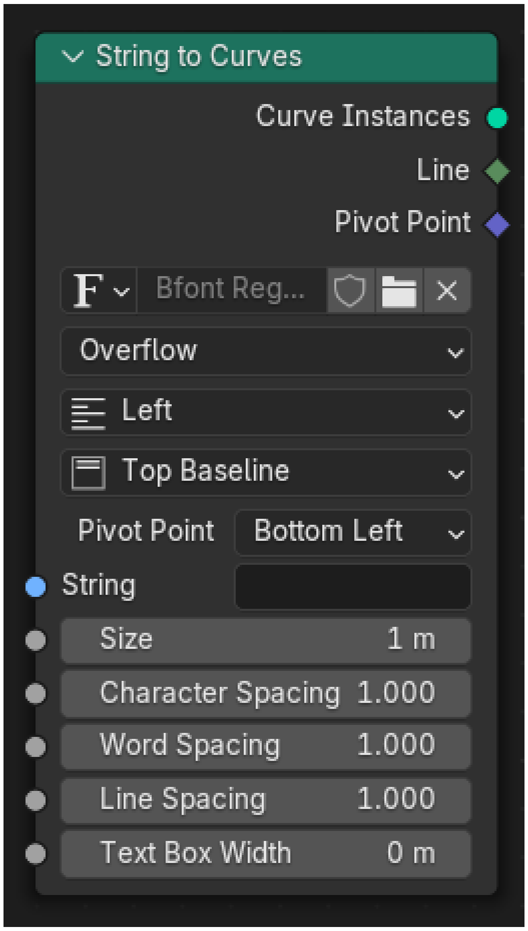
Outputs

Length
: Integer value representing the length of the input string.

String to Curves Node

The *String to Curves* converts a string to curve instances. Each unique character used in the string is converted to a curve once, and further uses of that character are more instances of the same geometry.

This makes processing the output geometry very efficient, because each unique character only has to be processed once. However, it means that the result will be the same for every instance of the same character. To process each character individually, the Realize Instances Node can be used.



Bfont Reg...
Overflow
Left
Top Baseline
Pivot Point Bottom Left
Size 1 m
Character Spacing 1.000
Word Spacing 1.000
Line Spacing 1.000
Text Box Width 0 m

Tip

Socket inspection can be used to see the value of the string input used when the node was evaluated, by holding the mouse over the socket.

Inputs

String
  Standard string input.

Size
  The size of each character. The values of the other inputs are scaled by this value.

Character Spacing

A factor by which the space between each character (kerning) is scaled on the X axis.

Word Spacing

A factor by which whitespace between words is scaled on the X axis.

Line Spacing

The distance between separate lines in the output. Scaled by the *Size* input.

Text Box Width

The maximum width of each line, though individual words will not be wrapped.

Text Box Height

The maximum height for all the lines of the text.

Properties

Font

Font glyph used to generate the curve.

Overflow

Overflow::	Wraps the text at the *Text Box Width*.
Scale To Fit::	Scales the text size to fit the *Text Box Width* and *Text Box Height*.
Truncate::	Only outputs text characters that fit within the width and height, based on the *Size* input. Any part of the string that did not fit is moved to the *Remainder* output.

Alignment

Left::	Aligns the text to the left.
Center::	Aligns the text to the center.
Right::	Aligns the text to the right.
Justify::	Aligns the text to the left and right.
Flush::	

	Aligns the text to the left and right with equal character spacing.

Align Y

Top::	Aligns the text to the top.
Top Baseline::	Aligns the text to the top baseline.
Middle::	Aligns the text to the middle.
Bottom Baseline::	Aligns the text to the bottom baseline.
Bottom::	Aligns the text to the bottom.

Pivot Point

Controls where on each character the output *Pivot Point* is placed.

Midpoint::	Place the pivot points at the center of each character's bounds.
Top Left::	Place the pivot points at the top left of each character's bounds.
Top Center::	Place the pivot points at the middle of the top of each character's bounds.
Top Right::	Place the pivot points at the top right of each character's bounds.
Bottom Left::	Place the pivot points at the bottom left of each character's bounds.
Bottom Center::	Place the pivot points at the middle of bottom of each character's bounds.
Bottom Right::	Place the pivot points at the bottom right of each character's bounds.

Outputs

Curve Instances

Curve instances geometry.

Remainder

The part of the text that did not fit in the box described by the *Text Box Height* and *Text Box Width* inputs. Only used in the *Truncate* overflow mode.

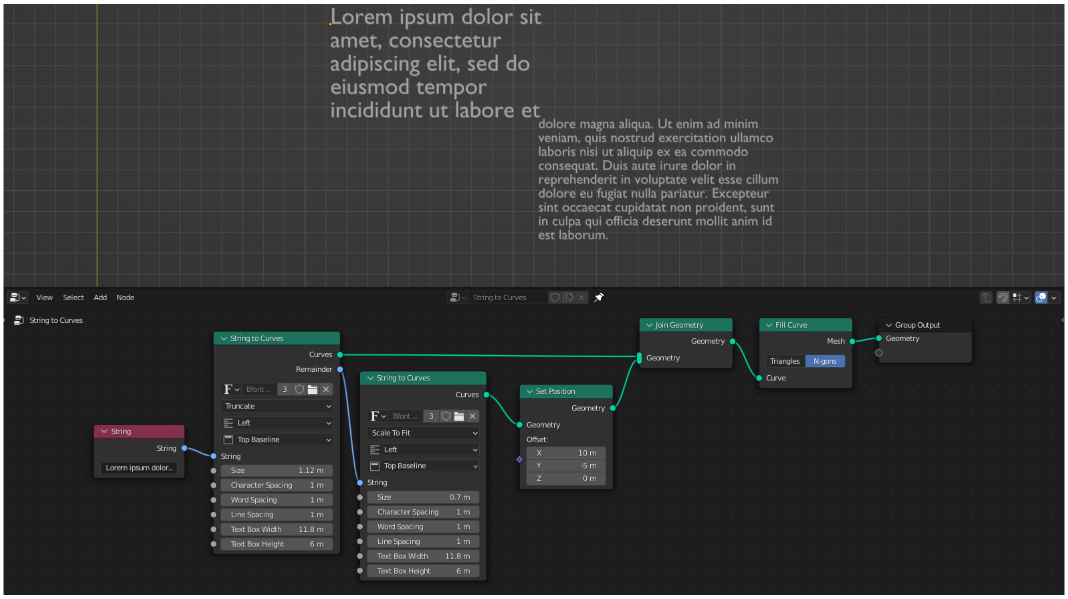
Line

An attribute field containing the line index of each character (on the instance domain).

Pivot Point

Outputs the position described by the *Pivot Point* drop-down in the local space of each instance.

Examples



Lorem ipsum dolor sit
amet, consectetur
adipiscing elit, sed do
eiusmod tempor
incididunt ut labore et

dolore magna aliqua. Ut enim ad minim
veniam, quis nostrud exercitation ullamco
laboris nisi ut aliquip ex ea commodo
consequat. Duis aute irure dolor in
reprehenderit in voluptate velit esse cillum
dolore eu fugiat nulla pariatur. Excepteur
sint occaecat cupidatat non proident, sunt
in culpa qui officia deserunt mollit anim id
est laborum.

String: Lorem ipsum dolor...

String to Curves: Truncate, Left, Top Baseline, Size 1.12 m, Character Spacing 1 m, Word Spacing 1 m, Line Spacing 1 m, Text Box Width 11.8 m, Text Box Height 6 m

String to Curves: Scale To Fit, Left, Top Baseline, Size 0.7 m, Character Spacing 1 m, Word Spacing 1 m, Line Spacing 1 m, Text Box Width 11.8 m, Text Box Height 6 m

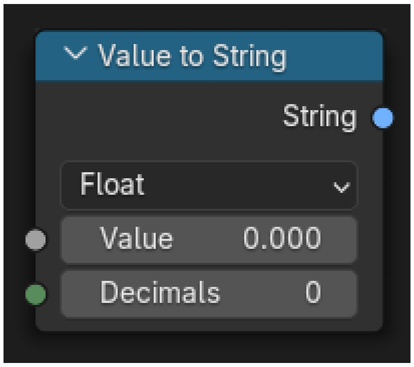
Set Position: Offset: X 10 m, Y -5 m, Z 0 m

Fill Curve: N-gons

View
Select
Add
Node
String to Curves
String to Curves
String
String
String to Curves
Curves
Remainder
Bfont ...
3
String
String to Curves
Curves
String
Set Position
Geometry
Geometry
Offset:
Join Geometry
Geometry
Geometry
Fill Curve
Mesh
Triangles
N-gons
Curve
Group Output
Geometry

The node can be used to make overflowing text boxes. Here, the text that does not fit into the first node’s fix-sized text box is passed to a separate *String to Curves* node. And finally added with a *Scale to Fit* node.

Value to String Node



The *Value to String* node generates string representation of the input value.

Inputs

Value
: Floating-point value to be converted.

Decimals *Float Data Type*
: Integer value used to determine the precision of the output value.

Properties

Data Type
: The type of numerical value to convert to a string.

Float::	Convert a floating-point value to a string.
Integer::	Convert a 32-bit integer to a string.

Outputs

String
: String value representation of the input.

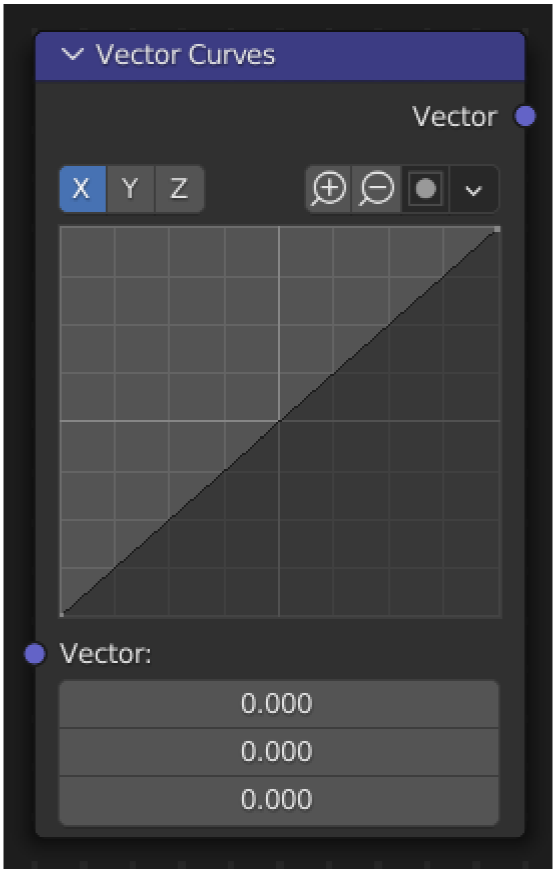
Vector Utility Nodes

Nodes for modifying vector quantities.

- Vector Curves Node
- Vector Math Node
- Vector Rotate Node

- Combine XYZ Node
- Mix Vector Node
- Separate XYZ Node

Vector Curves Node


Vector Curves
Vector
X
Y
Z
Vector:
0.000
0.000
0.000

The Vector Curves node maps an input vector components to a curve.

Use this curve node to slow things down or speed them up from the original scene.

Inputs

In the shader context the node also has an additional Factor property.

Factor
: Controls the amount of influence the node exerts on the output vector.

Vector
: Standard vector input.

Properties

Channel
X, Y, Z

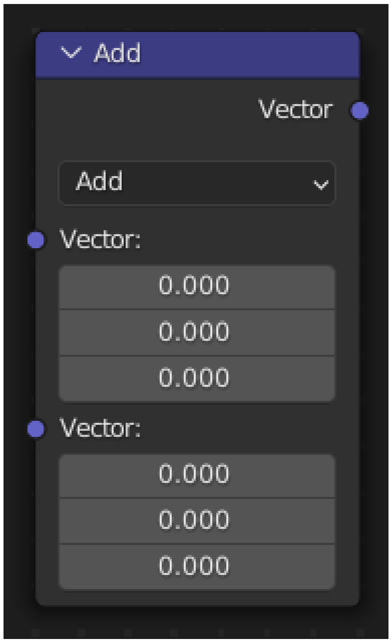
Curve
For the curve controls see: Curve widget.

Outputs

Vector
Standard vector output.

Vector Math Node

The *Vector Math* node performs the selected math operation on the input vectors.


Add
Vector
Add
Vector:
0.000
0.000
0.000
Vector:
0.000
0.000
0.000

Inputs

The inputs of the node are dynamic. Some inputs are only available in certain operations. For instance, the *Scale* input is only available in the *Scale* operator.

Vector

  Input vector $A = \begin{pmatrix} A_x \\ A_y \\ A_z \end{pmatrix}$.

Vector

  Input vector $B = \begin{pmatrix} B_x \\ B_y \\ B_z \end{pmatrix}$.

Scale

  Input Scale s.

Properties

Operation

The vector math operator to be applied on the input vectors.

Add::	The sum of A and B. \(\begin{pmatrix} A_x + B_x \\ A_y + B_y \\ A_z + B_z \end{pmatrix}\)
Subtract::	The difference between A and B. \(\begin{pmatrix} A_x - B_x \\ A_y - B_y \\ A_z - B_z \end{pmatrix}\)
Multiply::	The entrywise product of A and B. \(\begin{pmatrix} A_x \cdot B_x \\ A_y \cdot B_y \\ A_z \cdot B_z \end{pmatrix}\)
Divide::	The entrywise division of A by B. Division by zero results in zero. \(\begin{pmatrix} A_x / B_x \\ A_y / B_y \\ A_z / B_z \end{pmatrix}\)
Multiply Add::	The entrywise combination of the multiply and addition operations. \(A × B + C\)
Cross Product::	The cross product of A and B. \(\begin{pmatrix} A_y \cdot B_z - A_z \cdot B_y \\ A_z \cdot B_x - A_x \cdot B_z \\ A_x \cdot B_y - A_y \cdot B_x \end{pmatrix}\)
Project::	The projection of A onto B.
Reflect::	The reflection of A around the normal B. B need not be normalized.
Refract::	For a given incident vector A, surface normal B and ratio of indices of refraction (IOR), refract outputs the refraction vector R.
Faceforward::	Orients a vector A to point away from a surface B as defined by its normal C. Computes \((dot(B, C) < 0) ? A : -A\).
Dot Product::	The dot product of A and B. \(A_x \cdot B_x + A_y \cdot B_y + A_z \cdot B_z\)
Distance::	The distance between A and B.
Length::	The length of A. \(\sqrt{A_x^2 + A_y^2 + A_z^2}\)
Scale::	The result of multiplying A by the scalar input *Scale*. \(\begin{pmatrix} s \cdot A_x \\ s \cdot A_y \\ s \cdot A_z \end{pmatrix}\)

Normalize::	The result of normalizing A. The result vector points to the same direction as A and has a length of 1. If A is $(0, 0, 0)$, the result is $(0, 0, 0)$ as well.
Wrap::	The entrywise output of a value between Min and Max based on the absolute difference between the input value and the nearest integer multiple of Max less than the value.
Snap::	The result of rounding A to the largest integer multiple of B less than or equal A.
Floor::	Rounds the input value entrywise down to the nearest integer.
Ceil::	Rounds the input value entrywise up to the nearest integer.
Modulo::	The entrywise modulo of A by B.
Fraction::	Returns the fractional part of the *value* entrywise.
Absolute::	The entrywise absolute value of A.
Minimum::	The entrywise minimum value from A and B.
Maximum::	The entrywise maximum value from A and B.
Sine::	The entrywise Sine of A.
Cosine::	The entrywise Cosine of A.
Tangent::	The entrywise Tangent of A.

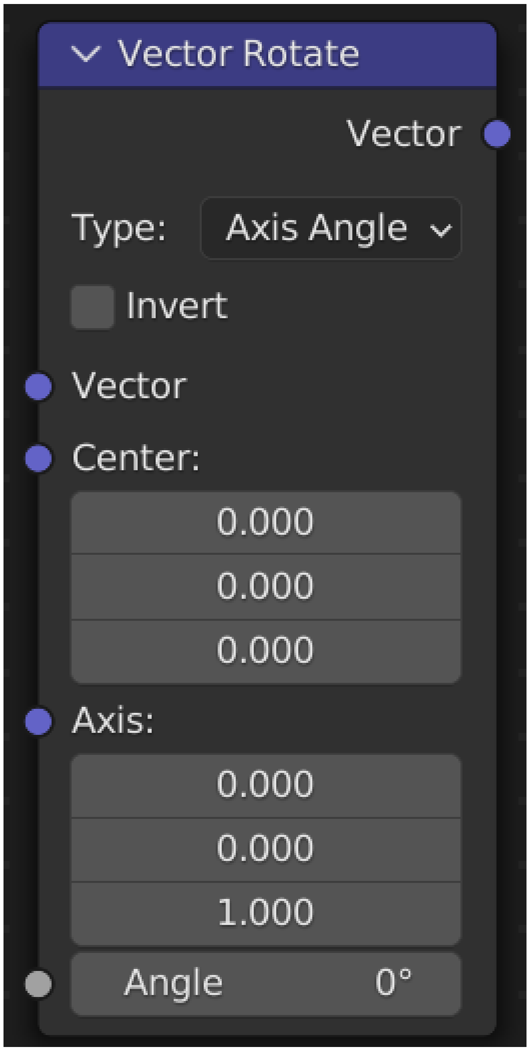
Outputs

The output of the node is dynamic. It is either a vector or a scalar depending on the operator. For instance, the *Length* operator has a scalar output while the *Add* operator has a vector output.

Vector
Output vector.

Value
Output value.

Vector Rotate Node



Type: Axis Angle

☐ Invert

Vector

Center:
0.000
0.000
0.000

Axis:
0.000
0.000
1.000

Angle 0°

The *Vector Rotate Node* provides the ability to rotate a vector around a pivot point (*Center*).

Inputs

Vector
: Vector to be rotated.

Center

Point to rotate around.

Axis

Axis to rotate around.

Angle

Angle to rotate the input vector by.

Rotation

When *Type* is set to *Euler*, rotate the input vector by these angles around the X, Y, then Z axes in that order.

Properties

Type

The type of angle input.

X/Y/Z Axis::	Rotates the vector around the defined axis and the amount of rotation is defined by the *Angle* input.
Axis Angle::	Rotates the vector around an arbitrary axis defined by the *Axis* input vector. The amount of rotation is defined by the *Angle* input.
Euler::	Rotates the vector about a center point defined by the *Center* input vector. The amount of rotation on each axis is defined by the *Rotation* input vector.

Invert

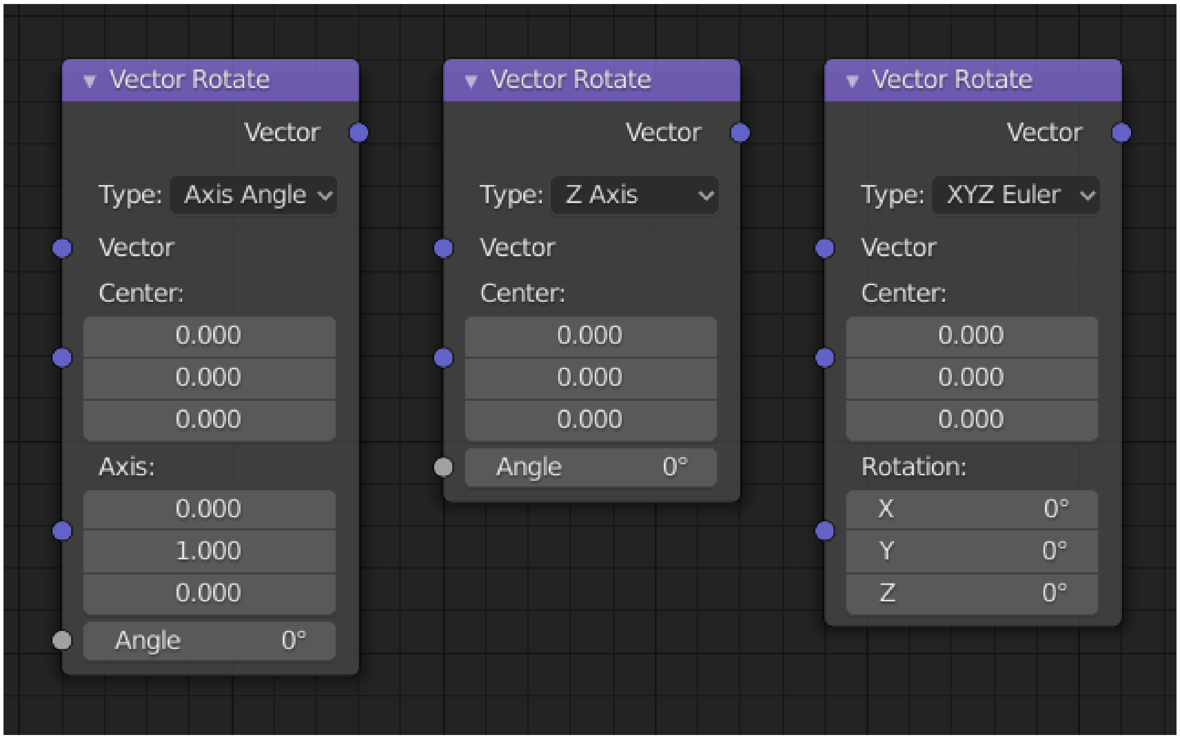
Inverts the rotation angle.

Outputs

Vector

The rotated vector.

Examples


Vector Rotate
Vector
Vector
Vector Rotate
Vector
Vector
Vector Rotate
Vector
Vector

Type: Axis Angle
Center: 0.000, 0.000, 0.000
Axis: 0.000, 1.000, 0.000
Angle 0°

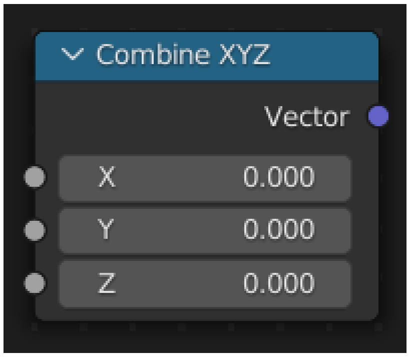
Type: Z Axis
Center: 0.000, 0.000, 0.000
Angle 0°

Type: XYZ Euler
Center: 0.000, 0.000, 0.000
Rotation: X 0°, Y 0°, Z 0°

Vector Rotate node example.

Combine XYZ Node

The *Combine XYZ Node* combines a vector from its individual components.


Combine XYZ
Vector
X 0.000
Y 0.000
Z 0.000

Inputs

- X
- Y
- Z

Properties

This node has no properties.

Output

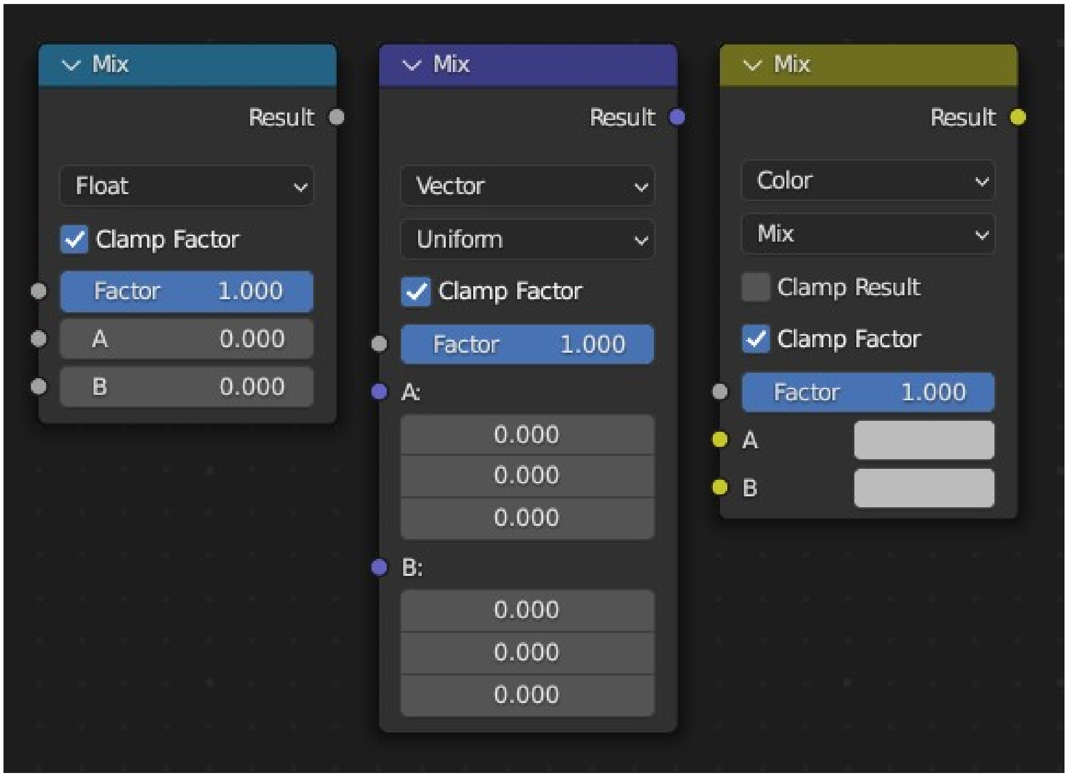
Vector
    Standard vector output.

Note

The vector is not normalized.

Mix Vector Node

The *Mix Node* mixes values, colors and vectors inputs using a factor to control the amount of interpolation. The *Color* mode has additional blending modes.


Mix
Result
Float
Clamp Factor
Factor 1.000
A 0.000
B 0.000
Mix
Result
Vector
Uniform
Clamp Factor
Factor 1.000
A:
0.000
0.000
0.000
B:
0.000
0.000
0.000
Mix
Result
Color
Mix
Clamp Result
Clamp Factor
Factor 1.000
A
B

Inputs

Factor
: Controls the amount of mixing between the A and B inputs.

A/B
: The two inputs that are mixed together.

Properties

Data Type

The data type that is used for mixing. The node supports float, vector, color, and rotation data types.

Factor Mode (Vector only)

The factor mode can be set to *Uniform* and *Non-Uniform*. In uniform mode, a single float controls the factor. In non-uniform mode, a vector controls the factor for each XYZ channel separately.

Mix (Color only)

The Blend modes can be selected in the select menu. See Color Blend Modes for details on each blending mode.

Add, Subtract, Multiply, Screen, Divide, Difference, Darken, Lighten, Overlay, Color Dodge, Color Burn, Hue, Saturation, Value, Color, Soft Light, Linear Light

Clamp Factor

Limit the factor value between 0.0 and 1.0. If this option is unchecked then the node operates using *Extrapolation*.

Clamp Result (Color only)

Limit the Result to the range between 0.0 and 1.0.

Outputs

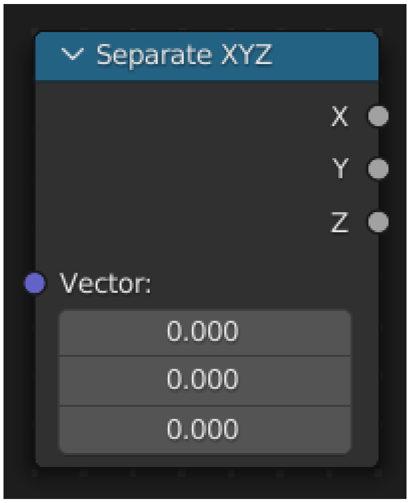
Result

Output the result of the mix using the data type selected.

Examples

See the Color > Mix page for additional examples: Mix Color Node

Separate XYZ Node


Separate XYZ
X
Y
Z
Vector:
0.000
0.000
0.000

The *Separate XYZ Node* splits a vector into its individual components.

Input

Vector
: Standard vector input.

Properties

This node has no properties.

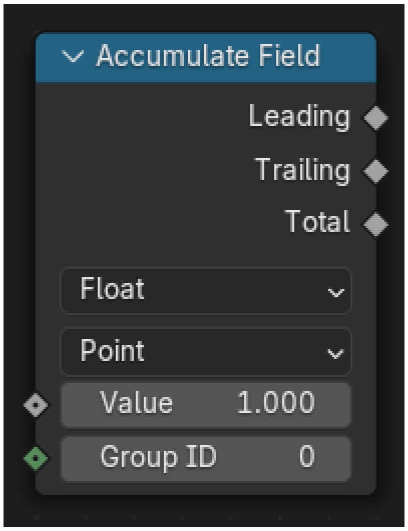
Outputs

- X
- Y
- Z

Field Utility Nodes

- Accumulate Field Node
- Evaluate at Index Node
- Evaluate on Domain Node

Accumulate Field Node



The *Accumulate Field* node counts a running total of its input values, in the order defined by the geometry's indices. The node's essential operation is just addition, but instead of only outputting the final total, it outputs the current value at every element.

Inputs

Value

The values to be accumulated.

Warning

When accumulating integer values, be careful to make sure that there are not too many large values. The maximum integer that Blender stores internally is around 2 billion. After that, values may wrap around and become negative. See wikipedia for more information.

Group Index

An index used to group values together for multiple separate accumulations. This can be thought of as a choice of the "bin" in which to place each value. This input has no effect when it is only a single value.

Properties

Data Type

Float::	The node will accumulate a *Float* field.
Integer::	The node will accumulate an *Integer* field.
Vector::	The node will accumulate a *Vector* field.
Transform::	The node will accumulate a *Matrix* field.

Domain

The attribute domain used for accumulation and for evaluation of the *Value* input.

Output

Leading

The running total of values in the corresponding group, starting at the first value.

Trailing

The running total of values in the corresponding group, starting at zero.

Total

The total of all the values in the corresponding group.

Examples

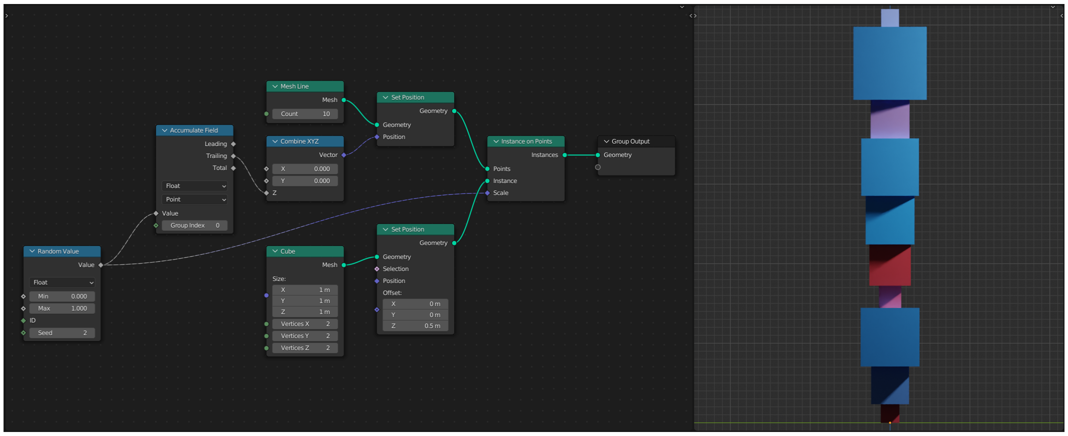
Table

Value	Group Index	Leading	Trailing	Total
1	7	1	0	6
3	7	4	1	6
2	7	6	4	6

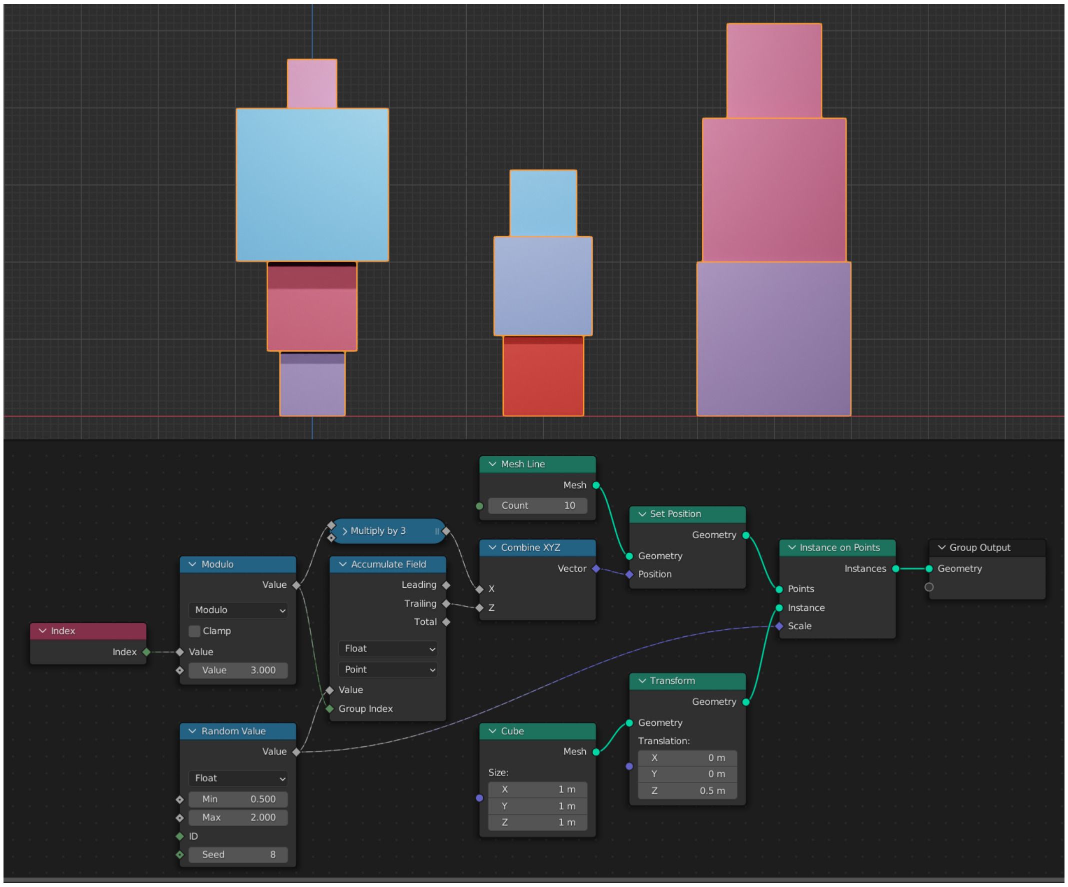
Value	Group Index	Leading	Trailing	Total
1	3	1	0	3
0	3	1	1	3
2	3	3	1	3

A few examples of input values and the node's results. One important takeaway from this table is that the specific values for the *Group Input* do not matter; it only matters that the values are shared between elements.

Stacking Boxes


Random Value
Value
Float
Min 0.000
Max 1.000
ID
Seed 2
Accumulate Field
Leading
Trailing
Total
Float
Point
Value
Group Index 0
Mesh Line
Mesh
Count 10
Combine XYZ
Vector
X 0.000
Y 0.000
Z
Cube
Mesh
Size:
X 1 m
Y 1 m
Z 1 m
Vertices X 2
Vertices Y 2
Vertices Z 2
Set Position
Geometry
Geometry
Position
Set Position
Geometry
Geometry
Selection
Position
Offset:
X 0 m
Y 0 m
Z 0.5 m
Instance on Points
Instances
Points
Instance
Scale
Group Output
Geometry

Here, the node is used in combination with the [Random Value Node]() to create a stack of randomly scaled boxes. The *Group Index* input is not used, because all boxes are meant to be in the same stack.


Mesh Line
Mesh
Multiply by 3
Modulo
Value
Accumulate Field
Leading
Trailing
Total
Combine XYZ
Vector
X
Z
Set Position
Geometry
Geometry
Position
Instance on Points
Instances
Points
Instance
Scale
Group Output
Geometry
Index
Index
Value
Value
Group Index
Random Value
Value
ID
Cube
Mesh
Transform
Geometry
Geometry

Mesh Line: Count 10

Modulo: Modulo, ☐ Clamp, Value 3.000

Accumulate Field: Float, Point

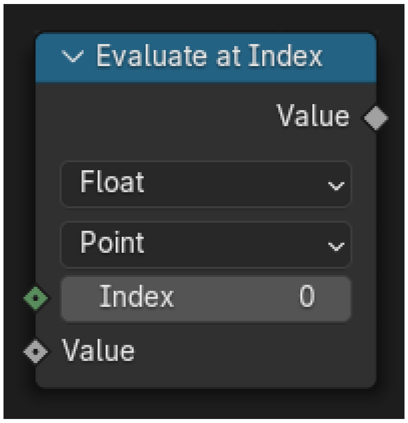
Random Value: Float, Min 0.500, Max 2.000, Seed 8

Cube: Size: X 1 m, Y 1 m, Z 1 m

Transform: Translation: X 0 m, Y 0 m, Z 0.5 m

A slightly more complicated version of the previous example, using the *Group Index* input to create three separate stacks.

Evaluate at Index Node



The *Evaluate at Index* node allows accessing data of other elements in the context geometry. It is similar to the Sample Index Node. The main difference is that this node does not require a geometry input, because the geometry from the field context is used.

This node is also similar to the Evaluate on Domain Node node, except that the value to retrieve from the specified domain is specified by an index rather than an automatic domain interpolation.

Inputs

Index

The index of the element in the selected domain to retrieve data from, i.e. “the fourth face”, or “the first control point”.

Value

The field to retrieve data from.

Properties

Domain

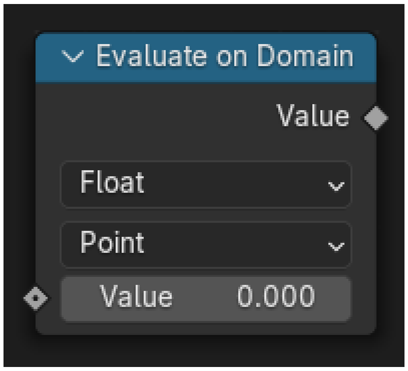
The attribute domain used for evaluation of the *Value* input. This is useful because it can be a different domain than the domain from the field context, i.e. to choose a vertex for each face.

Output

Value

The value of the input field at the given index.

Evaluate on Domain Node



The *Evaluate on Domain* allows evaluating a field for a different attribute domain than the domain from the field context. For example, the face index could be used instead of the face corner index, when setting the values of a UV Map

Note

This node is not necessary to retrieve data from other attribute domains; that is done automatically. Its utility comes from the fact that it's possible to control *when* the domain interpolation happens. Normally, input nodes interpolate their data to the current context's domain as soon as they create their output.

Tip

It may be preferable to use this node over the Capture Attribute Node, since it allows using a specific attribute domain without requiring a geometry socket input, which allows creating more reusable node groups.

See also

The method of retrieving data from another domain is somewhat similar to the Evaluate at Index Node.

Inputs

Value

The field to evaluate on the chosen attribute domain.

Properties

Domain

The attribute domain used for evaluation of the *Value* input.

Output

Value

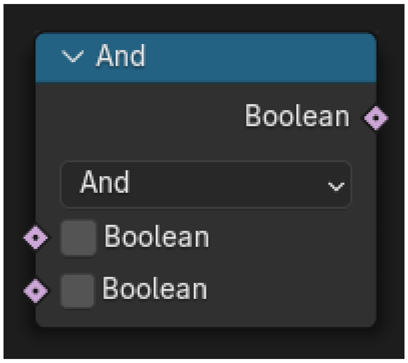
The values from the input, evaluated on the chosen domain, then interpolated to the domain from the field context.

Math Utility Nodes

- Boolean Math Node
- Clamp Node
- Compare Node
- Float Curve
- Float To Integer Node
- Hash Value Node
- Integer Math Node
- Map Range Node
- Math Node
- Mix Node

Boolean Math Node

The *Boolean Math* node performs a basic logical operation on its inputs.



Inputs

Boolean
Two standard Boolean inputs.

Properties

Mode

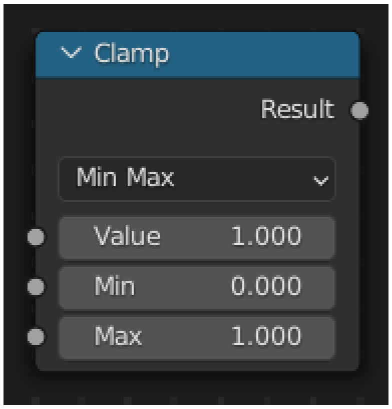
And::	True when both inputs are true. (AND)
Or::	True when at least one input is true. (OR)
Not::	Opposite of the input. (NOT)
Not And::	(True when at least one input is false. NAND)
Nor::	True when both inputs are false. (NOR)
Equal::	True when both inputs are equal. Also known as “exclusive nor”. (XNOR)
Not Equal::	(XOR) True when both inputs are different. Also known as “exclusive or”.
Imply::	True unless the first input is true and the second is false. (IMPLY)
Subtract::	True when the first input is true and the second is false. Also known as “”not imply”. (NIMPLY)

Output

Boolean
    Standard Boolean output.

Clamp Node

The *Clamp* node clamps a value between a minimum and a maximum.



Min Max

Value 1.000

Min 0.000

Max 1.000

Inputs

Value

The input value to be clamped.

Min

The minimum value.

Max

The maximum value.

Properties

Clamp Type

Method to clamp.

Min Max::	Constrain values between Min and Max.
Range::	Constrain values between Min and Max. When Min is greater than Max, constrain between Max and Min instead.

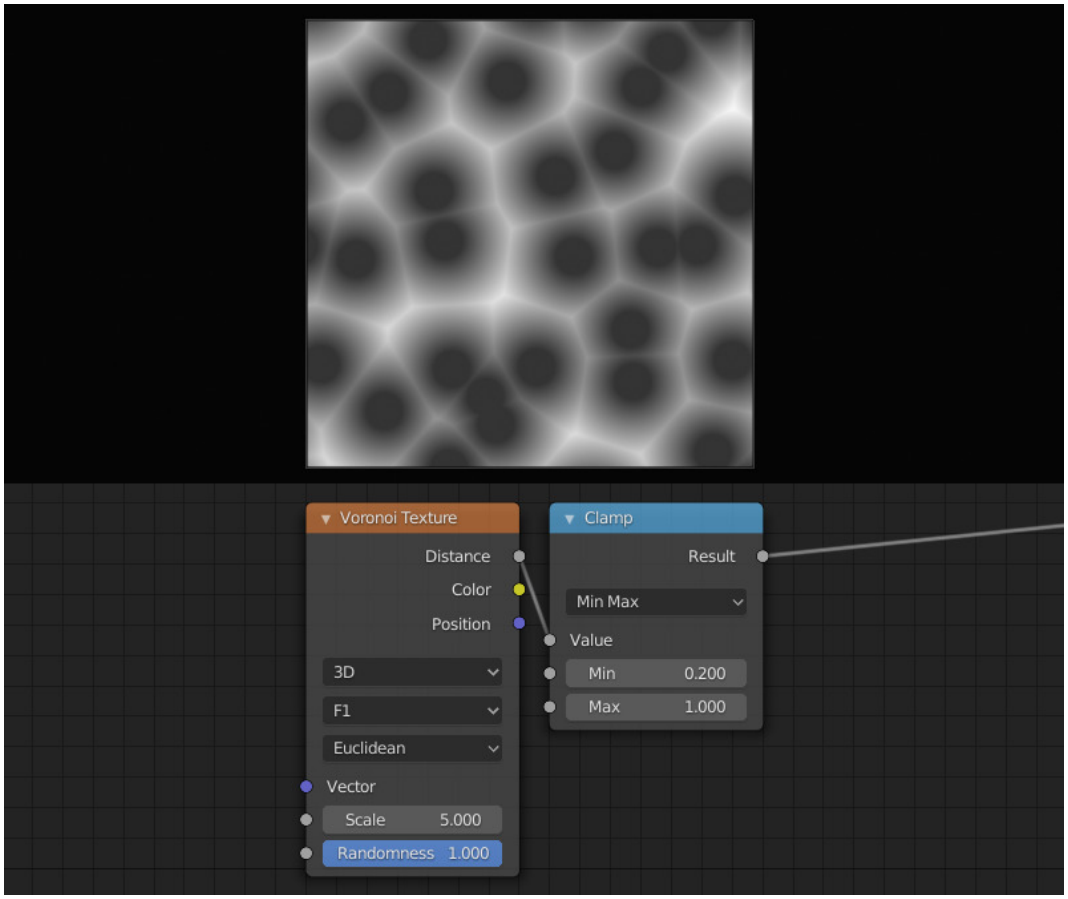
Outputs

Result

The input value after clamping.

Examples

The *Voronoi Texture* node outputs a value whose minimum is zero. We can use the *Clamp* node to clamp this value such that the minimum is 0.2.


Voronoi Texture
Distance
Color
Position
Vector
Clamp
Result
Value

3D

F1

Euclidean

Scale 5.000

Randomness 1.000

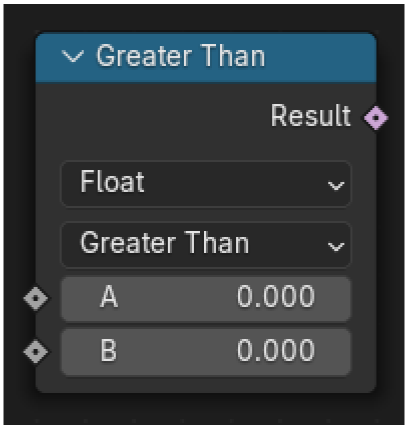
Min Max

Min 0.200

Max 1.000

Example of Clamp node.

Compare Node



The *Compare* node takes two inputs and does an operation to determine whether they are similar. The node can work on all generic data types, and has modes for vectors that contain more complex comparisons, which can help to reduce the number of necessary nodes, and make a node tree more readable.

Inputs

A, B

Standard value inputs of the selected type.

C

Compared against the dot product of two input vectors in when the *Mode* property is set to *Dot Product*.

Epsilon

This value is used as a threshold for still considering the two inputs as equal for the Equal and Not Equal operations.

Properties

Mode

Element-Wise::	Compare each axis of the input vectors separately, and output true only when the result is true for each axis.
Length::	Compare the length of the two input vectors.
Average::	Compare the average of the elements of the input vectors. This is the same as the implicit conversion used when setting the node's data type to *Float*.
Dot Product::	Compare the dot product of the two vectors with the separate *C* input, using the selected operation. The dot product outputs a single value that says how much the two vectors "agree".
Direction::	Compare the angle between the two vectors with the separate *Angle* input, using the selected operation. The vectors are normalized, so their length does not matter.

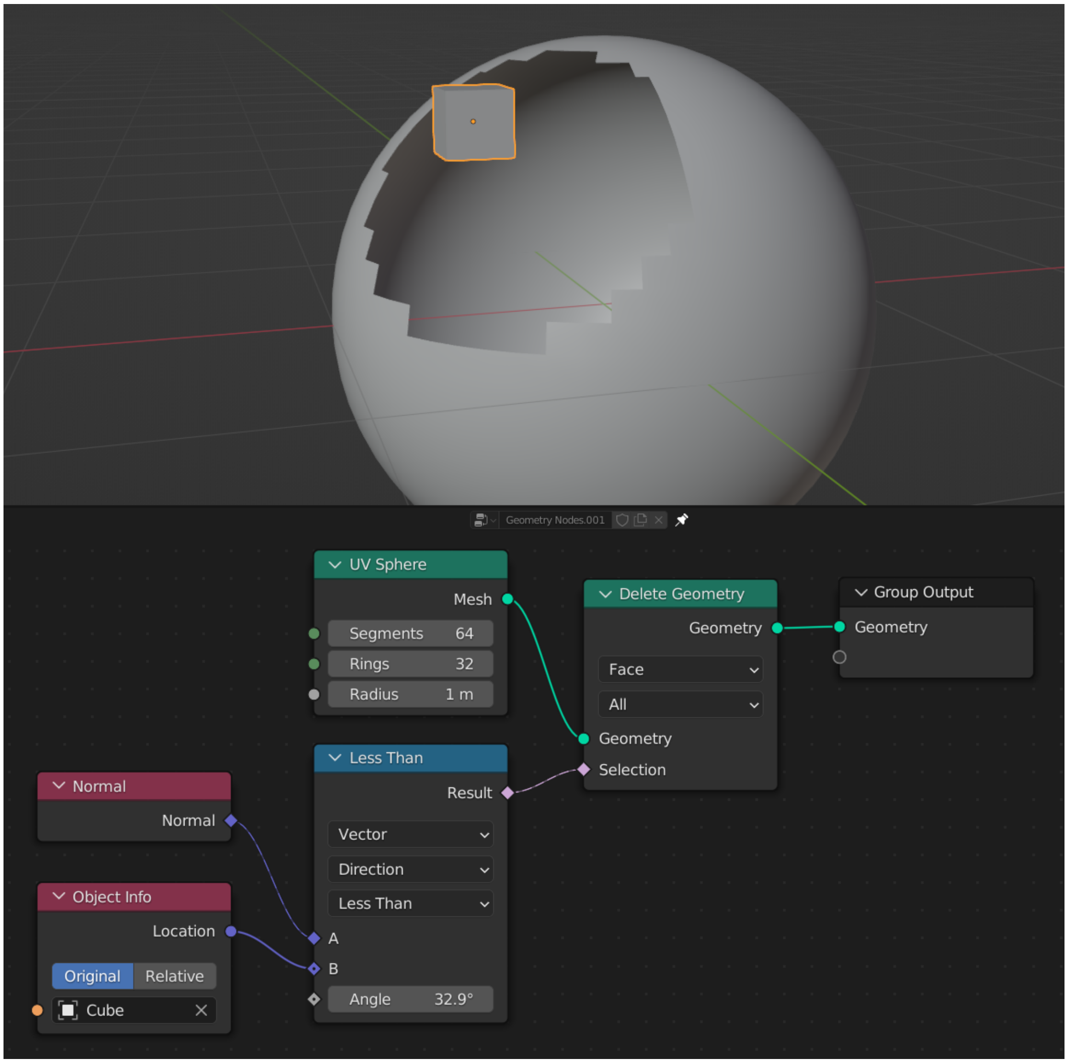
Operation

Less Than::	True when the first input is smaller than second input.
Less Than or Equal::	True when the first input is smaller than the second input or equal.
Greater Than::	True when the first input is greater than the second input.
Greater Than or Equal::	True when the first input is greater than the second input or equal.
Equal::	True when both the difference between the two inputs is smaller than the *Epsilon* input.
Not Equal::	True when both the difference between the two inputs is larger than the *Epsilon* input.
Brighter::	True when the first color input is brighter than the second.
Darker::	True when the first color input is darker than the second.

Output

Result
    Standard Boolean output.

Examples

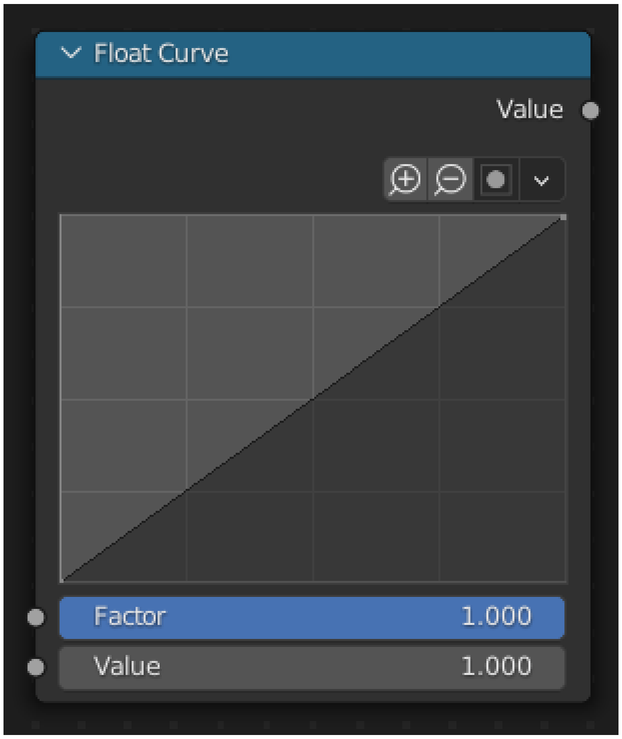

Geometry Nodes.001
UV Sphere
Mesh
Delete Geometry
Geometry
Group Output
Geometry
Geometry
Selection
Less Than
Result
Normal
Normal
Object Info
Location
A
B

Segments 64
Rings 32
Radius 1 m
Face
All
Vector
Direction
Less Than
Angle 32.9°
Original | Relative
Cube

Here, the compare node is used with the *Direction* mode to compare the direction of the sphere’s face normals to the “direction” of the cube object’s location. Anywhere that the directions are less than 32.9 degrees apart, the faces will be selected, and deleted.

Float Curve

The Float Curve node maps an input float to a curve and outputs a float value.


Float Curve
Value
Factor 1.000
Value 1.000

Inputs

Factor
: Controls the amount of influence the node exerts on the output value.

Value
: Standard float input.

Properties

Curve
: For the curve controls see: Curve widget.

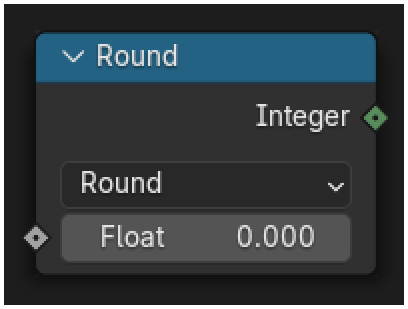
Outputs

Float

Standard float output.

Float To Integer Node

The *Float To Integer* node takes a single floating point number input and converts it to an integer with a choice of methods.



Inputs

Float
    Standard float value input.

Properties

Rounding Mode

Round::	Outputs the closest integer to Float, rounding either up or down based on the value.
Floor::	Outputs the closest integer less than Float, always rounding down.
Ceiling::	Outputs the closest integer greater than Float, always rounding up.
Truncate::	Outputs the closest integer between Float and zero. For positive numbers, acts like Floor. For negative numbers, acts as Ceiling.

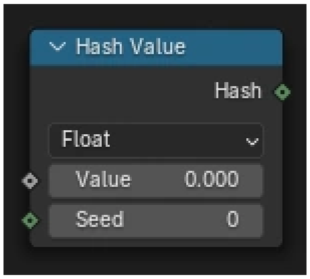
Output

Result
    Standard integer output.

Examples

Input Value	Round	Floor	Ceiling	Truncate
-69.6574	-70	-70	-69	-69
-3.14159	-3	-4	-3	-3
-1.5	-2	-2	-1	-1
1.5	2	1	2	1
3.14159	3	3	4	3
69.6574	70	69	70	69

Hash Value Node



The *Hash Value* node takes a value input and hashes this to an integer.

Important

Hashes cannot be relied upon to be used as unique identifiers because they are not guaranteed to be unique. It can be used to generate somewhat stable randomness especially in cases where *White Noise* does not offer enough flexibility.

Inputs

Value
: Value input determined by *Data Type* property.

Seed
: Integer input used to generate different Hashes .

Properties

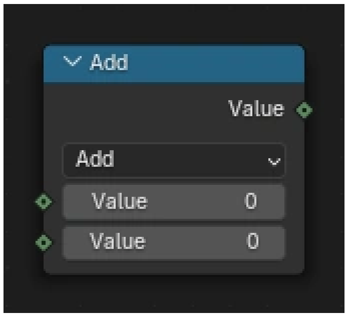
Data Type
: The data type that is used for the *Value* input. The node supports float, integer, vector, color, boolean, rotation, matrix and string data types.

Output

Hash
: Standard integer output.

Integer Math Node

The *Integer Math* node performs math operations.



Inputs

Value

Standard integer value input. Depending on the operation there will be more than one input.

Properties

Operation

The mathematical operator to be applied to the input values:

Functions

Add::	The sum of the two values.
Subtract::	The difference between the two values.
Multiply::	The product of the two values.
Divide::	The division of the first value by the second value.
Multiply Add::	The sum of the product of the two values with *Addend*.
Absolute::	The input value is read without regard to its sign. This turns negative values into positive values.
Negate::	Changes the sign of the input value.
Power::	The *Base* raised to the power of *Exponent*.

Comparison

Minimum::	Outputs the smallest of the input values.
Maximum::	Outputs the largest of two input values.
Sign::	Extracts the sign of the input value. All positive numbers will output 1. All negative numbers will output -1. And 0 will output 0.

Rounding

Divide Round::	Divide the values and round the result toward zero.
Divide Floor::	Divide the values and floor the result down to the nearest integer.
Divide Ceiling::	Divide the values and ceil the result up to the nearest integer .
Floored Modulo::	Returns the positive remainder of a division operation.
Modulo::	Outputs the remainder once the first value is divided by the second value.
Greatest Common Divisor::	The largest positive integer that divides into each of the values.
Least Common Multiple::	The smallest positive integer that is divisible by both values.

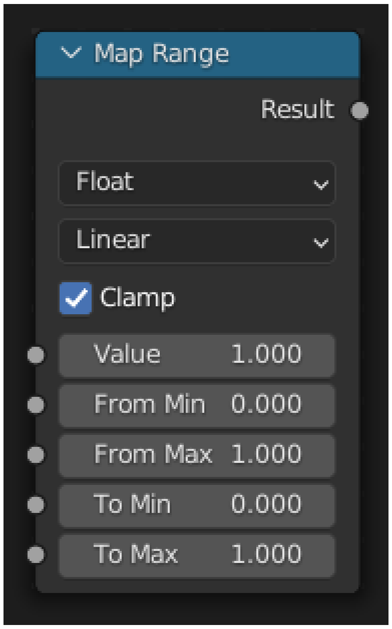
Output

Value

Standard integer output.

Map Range Node

The *Map Range* node remaps a value from a range to a target range.



Float

Linear

☑ Clamp

Value 1.000

From Min 0.000

From Max 1.000

To Min 0.000

To Max 1.000

Inputs

Value/Vector
: The input value or vector to be remapped.

From Min
: The lower bound of the range to remap from.

From Max
: The higher bound of the range to remap from.

To Min
: The lower bound of the target range.

To Max
: The higher bound of the target range.

Steps

The number of values allowed between *To Min* and *To Max* when using *Stepped Linear* interpolation. A higher value will give a smoother interpolation while lower values will progressively quantize the input.

Properties

Data Type

Map Range supports both Float and Vector data types. Changing the data type will also update the sockets to reflect the data type chosen.

Interpolation Type

The mathematical method used to transition between gaps in the numerical inputs.

Linear::	Linear interpolation between From Min and From Max values.
Stepped Linear::	Stepped linear interpolation between From Min and From Max values.
Smooth Step::	Smooth Hermite edge interpolation between From Min and From Max values.
Smoother Step::	Smoother Hermite edge interpolation between From Min and From Max values.

Clamp

If enabled, the output is clamped to the target range.

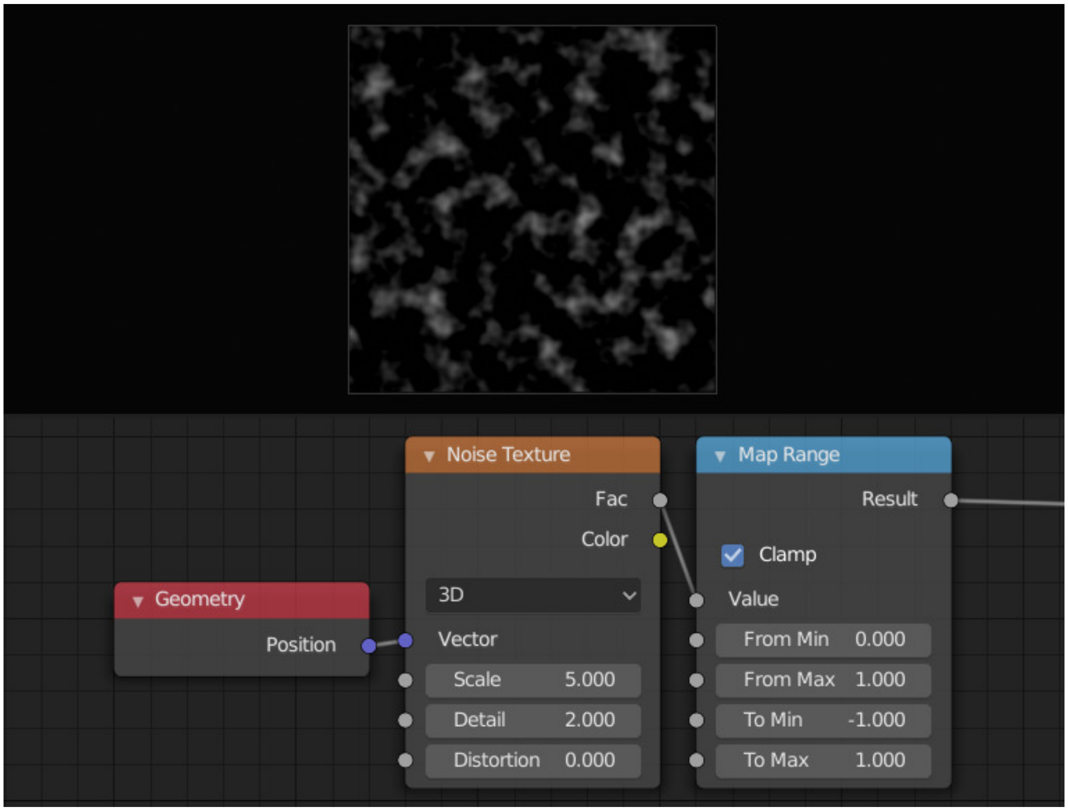
Outputs

Result/Vector

The input value after remapping.

Examples

The *Noise Texture* node outputs a value in the range [0, 1]. We can use the *Map Range* node to remap this value into the range [-1, 1].


Geometry
Position
Noise Texture
Fac
Color
Vector
Map Range
Result
Value

3D

Scale 5.000

Detail 2.000

Distortion 0.000

☑ Clamp

From Min 0.000

From Max 1.000

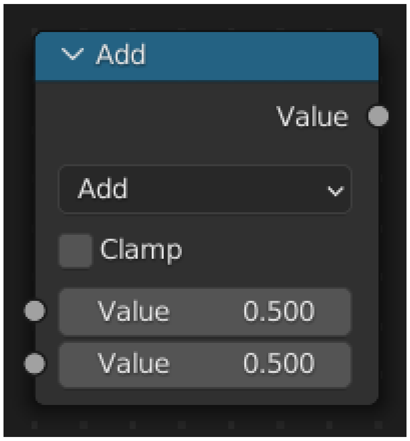
To Min -1.000

To Max 1.000

Example of Map Range node.

Math Node

The *Math Node* performs math operations.



Inputs

The inputs of the node are dynamic. Some inputs are only available for certain operations. For instance, the *Addend* input is only available for the *Multiply Add* operator.

Value
: Input Value. Trigonometric functions read this value as radians.

Addend
: Input Addend.

Base
: Input Base.

Exponent
: Input Exponent.

Epsilon
: Input Epsilon.

Distance

Input Distance.

Min
Input Minimum.

Max
Input Maximum.

Increment
Input Increment.

Scale
Input Scale.

Degrees
Input Degrees.

Radians
Input Radians.

Properties

Operation
The mathematical operator to be applied to the input values:

Functions

Add::	The sum of the two values.
Subtract::	The difference between the two values.
Multiply::	The product of the two values.
Divide::	The division of the first value by the second value.
Multiply Add::	The sum of the product of the two values with *Addend*.
Power::	The *Base* raised to the power of *Exponent*.
Logarithm::	The log of the value with a *Base* as its base.
Square Root::	The square root of the value.
Inverse Square Root::	

One divided by the square root of the value.

Absolute:: The input value is read without regard to its sign. This turns negative values into positive values.

Exponent:: Raises Euler’s number to the power of the value.

Comparison

Minimum:: Outputs the smallest of the input values.

Maximum:: Outputs the largest of two input values.

Less Than:: Outputs 1.0 if the first value is smaller than the second value. Otherwise the output is 0.0.

Greater Than:: Outputs 1.0 if the first value is larger than the second value. Otherwise the output is 0.0.

Sign:: Extracts the sign of the input value. All positive numbers will output 1.0. All negative numbers will output -1.0. And 0.0 will output 0.0.

Compare:: Outputs 1.0 if the difference between the two input values is less than or equal to *Epsilon*.

Smooth Minimum:: Smooth Minimum.

Smooth Maximum::

Smooth Maximum.

Rounding

Round:: Rounds the input value to the nearest integer.

Floor:: Rounds the input value down to the nearest integer.

Ceil:: Rounds the input value up to the nearest integer.

Truncate:: Outputs the integer part of the *value*.

Fraction:: Returns the fractional part of the *value*.

Truncated Modulo::

Outputs the remainder once the first value is divided by the second value.

Floored Modulo:: Returns the positive remainder of a division operation.

Wrap::

Outputs a value between *Min* and *Max* based on the absolute difference between the input value and the nearest integer multiple of *Max* less than the value.

Snap:: Rounds the input value down to the nearest integer multiple of *Increment*.

Ping-pong:: Bounces back and forth between 0.0 and the *Scale* as the input value increases.

Trigonometric

Sine:: The Sine of the input value.

Cosine:: The Cosine of the input value.

Tangent:: The Tangent of the input value.

Arcsine:: The Arcsine of the input value.

Arccosine:: The Arccosine of the input value.

Arctangent:: The Arctangent of the input value.

Arctan2:: Outputs the Inverse Tangent of the first value divided by the second value measured in radians.

Hyperbolic Sine:: The Hyperbolic Sine of the input value.

Hyperbolic Cosine:: The Hyperbolic Cosine of the input value.

Hyperbolic Tangent:: The Hyperbolic Tangent of the input value.

Conversion

To Radians:: Converts the input from degrees to radians.

To Degrees:: Converts the input from radians to degrees.

Clamp

Limits the output to the range (0.0 to 1.0). See Clamp.

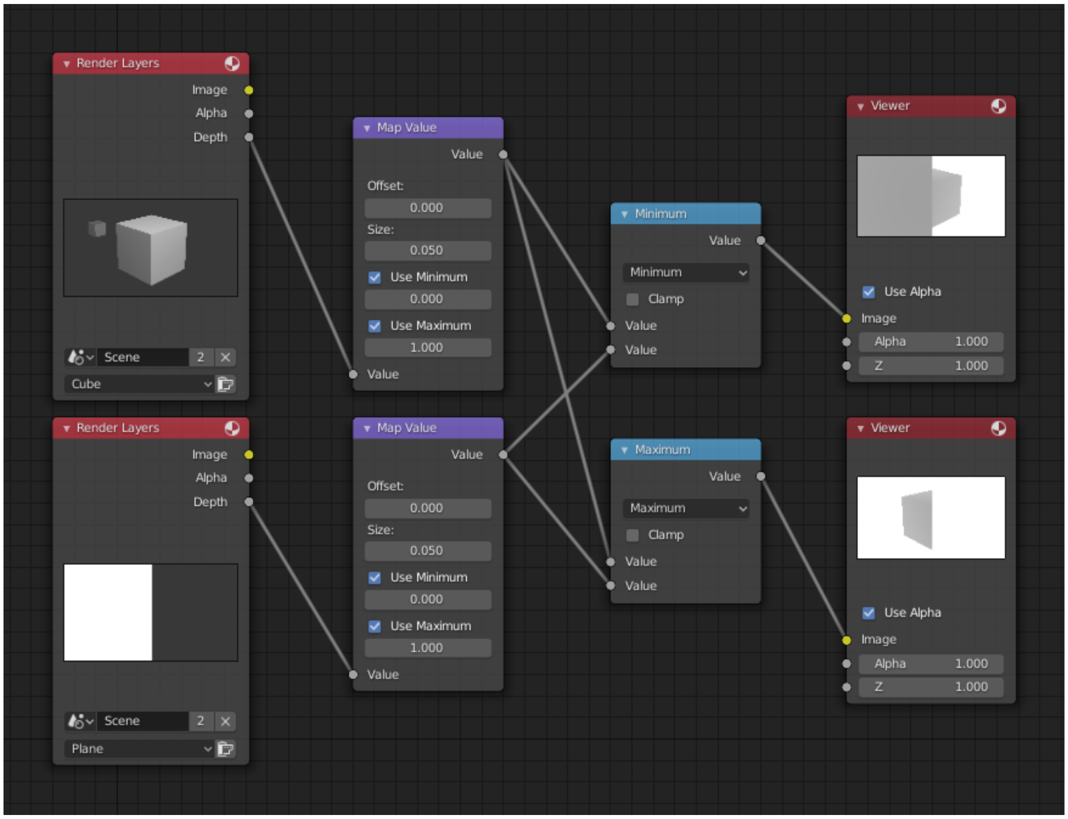
Outputs

Value

Numerical value output.

Examples

Manual Z-Mask


Render Layers
Image
Alpha
Depth
Scene
2
Cube
Map Value
Value
Offset:
0.000
Size:
0.050
Use Minimum
0.000
Use Maximum
1.000
Value
Minimum
Value
Minimum
Clamp
Value
Value
Viewer
Use Alpha
Image
Alpha 1.000
Z 1.000
Render Layers
Image
Alpha
Depth
Scene
2
Plane
Map Value
Value
Offset:
0.000
Size:
0.050
Use Minimum
0.000
Use Maximum
1.000
Value
Maximum
Value
Maximum
Clamp
Value
Value
Viewer
Use Alpha
Image
Alpha 1.000
Z 1.000

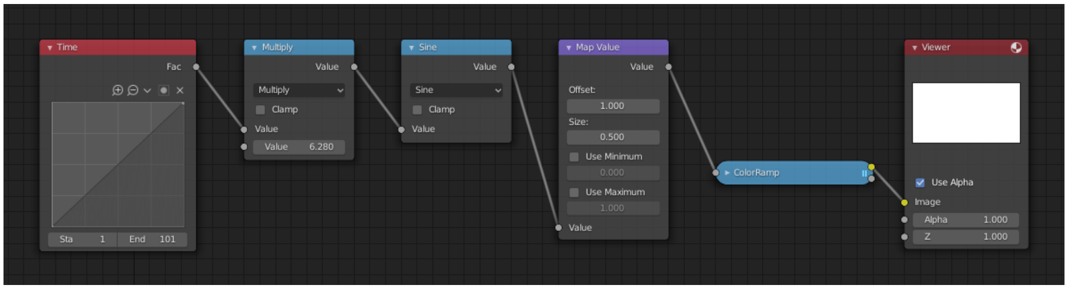
Minimum and maximum function example.

The top *Render Layers* node has a cube that is about 10 units from the camera. The bottom *Render Layers* node has a plane that covers the left half of the view and is 7 units from the camera. Both are fed through their respective *Map Value* nodes to multiply the depth value by 0.05 and clamp it to [0.0, 1.0], bringing it into a suitable range for displaying it as a color.

The Minimum node selects the smallest of the two depth values at each pixel. In the left half, it chooses the plane (because it’s closer than the cube), and in the right half, it chooses the cube (because it’s closer than the background, which is infinitely far away).

The Maximum node selects the largest of the two depth values at each pixel. In the left half, it chooses the cube (because it’s farther away than the plane), and in the right half, it chooses the background (because it’s farther away than the cube).

Using Sine Function to Pulsate


Time
Fac
Sta 1
End 101
Multiply
Value
Multiply
Clamp
Value
Value 6.280
Sine
Value
Sine
Clamp
Value
Map Value
Value
Offset:
1.000
Size:
0.500
Use Minimum
0.000
Use Maximum
1.000
Value
ColorRamp
Viewer
Use Alpha
Image
Alpha 1.000
Z 1.000

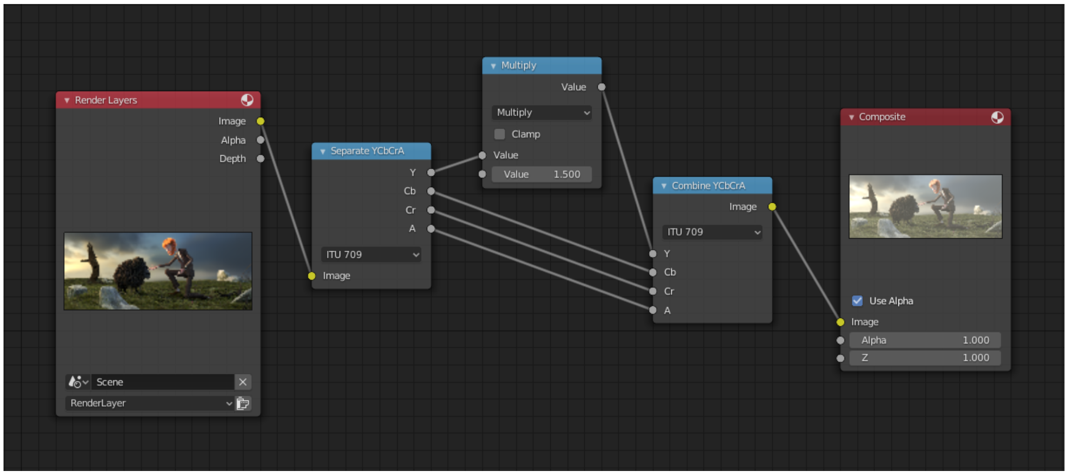
Using sine function example.

This example has a *Time* node putting out a linear sequence from 0 to 1 over the course of 101 frames. At frame 25, the output value is 0.25. That value is multiplied by 2 × pi (6.28) and converted to 1.0 by the Sine function, since \(sin(2 × pi/ 4) = sin(pi/ 2) = +1.0\).

Since the sine function can output values between (-1.0 to 1.0), the *Map Value* node scales that to 0.0 to 1.0 by taking the input (-1 to 1), adding 1 (making 0 to 2), and multiplying the result by 0.5 (thus scaling the output between 0 to 1). The default *Color Ramp* converts those values to a gray-scale. Thus, medium gray corresponds to a 0.0 output by the sine, black to -1.0, and white to 1.0. As you can see, \(sin(pi/ 2) = 1.0\). Like having your own visual color calculator! Animating this node setup provides a smooth cyclic sequence through the range of grays.

Use this function to vary, for example, the alpha channel of an image to produce a fading in/out effect. Alter the Z channel to move a scene in/out of focus. Alter a color channel value to make a color “pulse”.

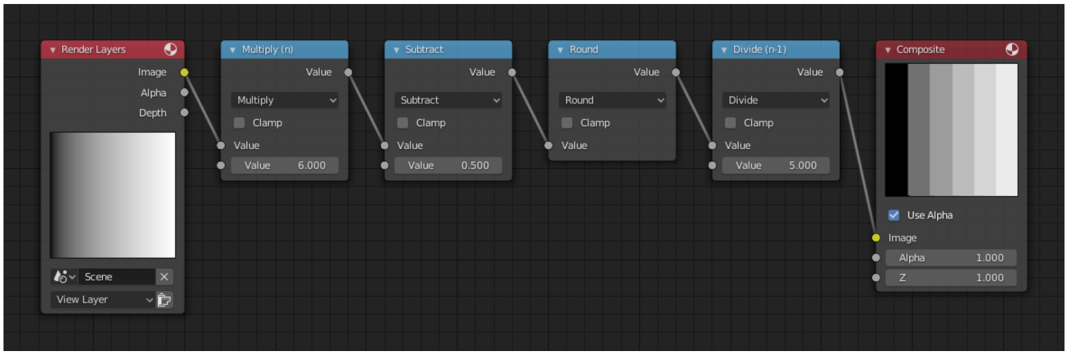
Brightening (Scaling) a Channel


Render Layers
Image
Alpha
Depth
Scene
RenderLayer
Separate YCbCrA
Y
Cb
Cr
A
ITU 709
Image
Multiply
Value
Multiply
Clamp
Value
Value 1.500
Combine YCbCrA
Image
ITU 709
Y
Cb
Cr
A
Composite
Use Alpha
Image
Alpha 1.000
Z 1.000

Scaling a channel example.

This example has a *Math (Multiply)* node increasing the luminance channel (Y) of the image to make it brighter. Note that you should use a *Map Value node* with min() and max() enabled to clamp the output to valid values. With this approach, you could use a logarithmic function to make a high dynamic range image. For this particular example, there is also a *Brightness/Contrast node* that might give simpler control over brightness.

Restrict Color Selection (Posterization)


Render Layers
Image
Alpha
Depth
Scene
View Layer
Multiply (n)
Value
Multiply
Clamp
Value
Value 6.000
Subtract
Value
Subtract
Clamp
Value
Value 0.500
Round
Value
Round
Clamp
Value
Divide (n-1)
Value
Divide
Clamp
Value
Value 5.000
Composite
Use Alpha
Image
Alpha 1.000
Z 1.000

Posterization example.

In this example, we restrict the color values to be one of the six values: 0, 0.2, 0.4, 0.6, 0.8, 1.

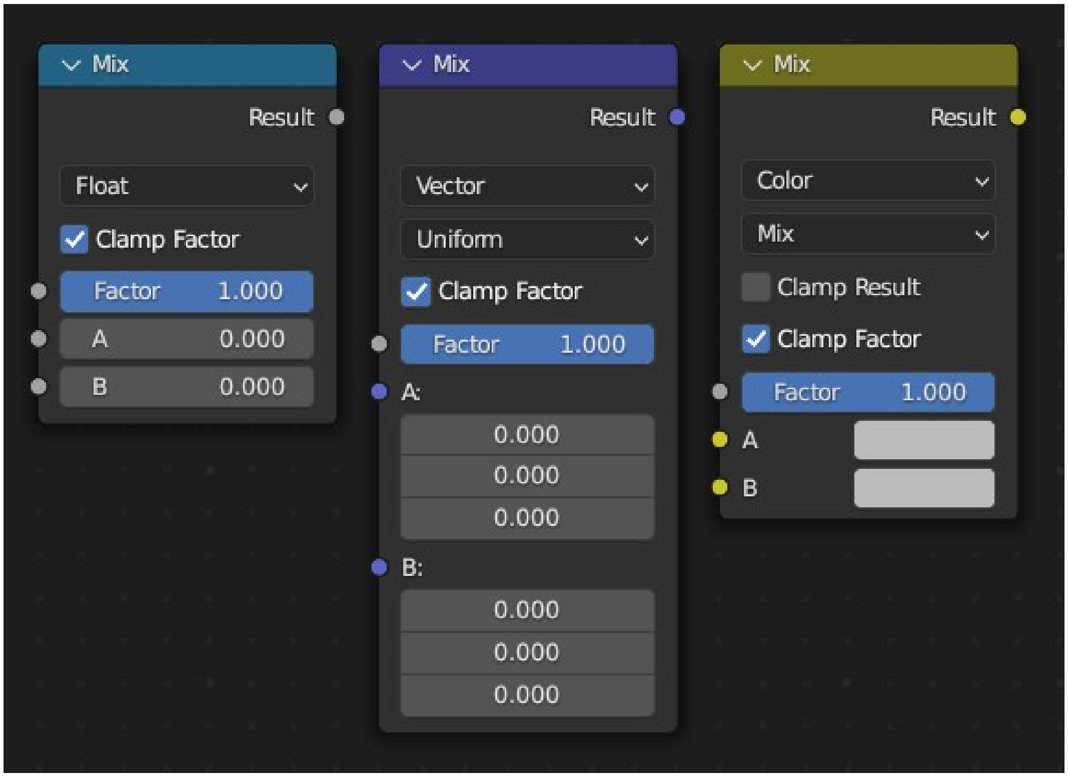
To split up a continuous range of values between 0 and 1 to certain set of values, the following function is used: $round(x \times n - 0.5) / (n - 1)$, where “n” is the number of possible output values, and “x” is the input pixel color. Read more about this function.

To implement this function in Blender, consider the node setup above. We string the Math nodes into a function that takes each color (values from 0 to 1), multiplies it up by six, the desired number of divisions (values become from 0 to 6), offsets it by 0.5 (-0.5 to 5.5), rounds the value to the nearest whole number (produces 0, 1, 2, 3, 4, 5), and then divides the image pixel color by five (0.0, 0.2, 0.4, 0.6, 0.8, 1.0).

In the case of a color image, you need to split it into separate RGB channels using *Separate/Combine Color* nodes and perform this operation on each channel independently.

Mix Node

The *Mix Node* mixes values, colors and vectors inputs using a factor to control the amount of interpolation. The *Color* mode has additional blending modes.


Mix
Result
Float
☑ Clamp Factor
Factor 1.000
A 0.000
B 0.000
Mix
Result
Vector
Uniform
☑ Clamp Factor
Factor 1.000
A:
0.000
0.000
0.000
B:
0.000
0.000
0.000
Mix
Result
Color
Mix
☐ Clamp Result
☑ Clamp Factor
Factor 1.000
A
B

Inputs

Factor
: Controls the amount of mixing between the A and B inputs.

A/B
: The two inputs that are mixed together.

Properties

Data Type

The data type that is used for mixing. The node supports float, vector, color, and rotation data types.

Factor Mode (Vector only)

The factor mode can be set to *Uniform* and *Non-Uniform*. In uniform mode, a single float controls the factor. In non-uniform mode, a vector controls the factor for each XYZ channel separately.

Mix (Color only)

The Blend modes can be selected in the select menu. See Color Blend Modes for details on each blending mode.

Add, Subtract, Multiply, Screen, Divide, Difference, Darken, Lighten, Overlay, Color Dodge, Color Burn, Hue, Saturation, Value, Color, Soft Light, Linear Light

Clamp Factor

Limit the factor value between 0.0 and 1.0. If this option is unchecked then the node operates using *Extrapolation*.

Clamp Result (Color only)

Limit the Result to the range between 0.0 and 1.0.

Outputs

Result

Output the result of the mix using the data type selected.

Examples

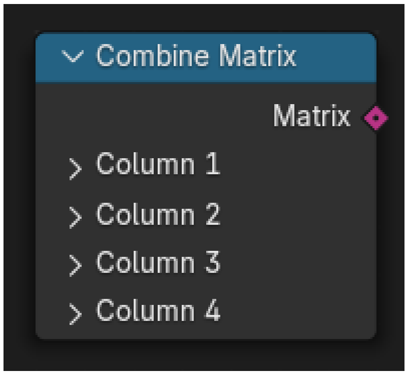
See the Color > Mix page for additional examples: Mix Color Node

Matrix Utility Nodes

- Combine Matrix Node
- Combine Transform Node
- Invert Matrix Node
- Matrix Determinant Node
- Multiply Matrices Node
- Project Point Node
- Separate Matrix Node
- Separate Transform Node
- Transform Direction Node
- Transform Point Node
- Transpose Matrix Node

Combine Matrix Node

The *Combine Matrix* node constructs a 4x4 matrix from its individual values.


Combine Matrix
Matrix
Column 1
Column 2
Column 3
Column 4

Inputs

The inputs of this node are split into panels for each column of the matrix. Each panel, has four value inputs for the four rows of the matrix.

Properties

This node has no properties.

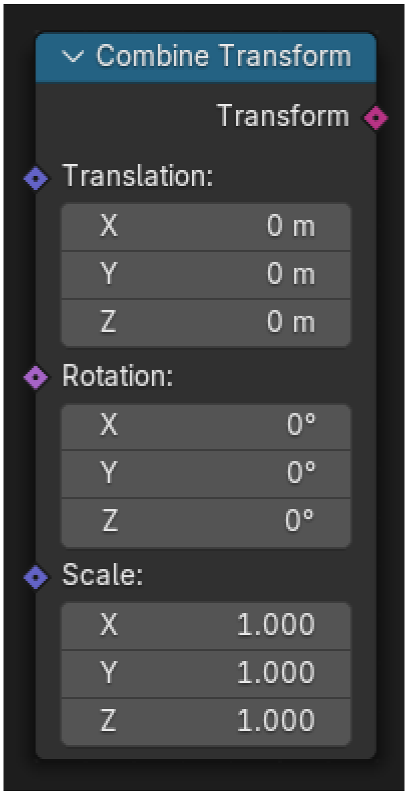
Outputs

Matrix

> The constructed matrix.

Combine Transform Node

The *Combine Transform* node combines a translation vector, a rotation vector, and a scale vector into a Transformation Matrix.


Combine Transform
Transform
Translation:
X 0 m
Y 0 m
Z 0 m
Rotation:
X 0°
Y 0°
Z 0°
Scale:
X 1.000
Y 1.000
Z 1.000

Inputs

Translation
: The translation vector.

Rotation
: The rotation vector.

Scale
: The scale vector.

Properties

This node has no properties.

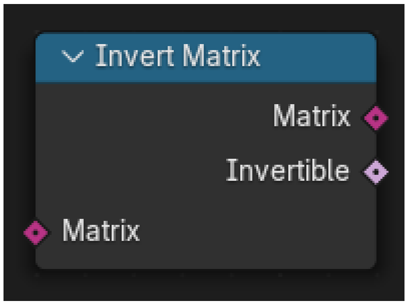
Outputs

Transform

The combined transformation matrix.

Invert Matrix Node

Returns the inverse of the given matrix.



Inputs

Matrix
: The matrix to invert.

Properties

This node has no properties.

Outputs

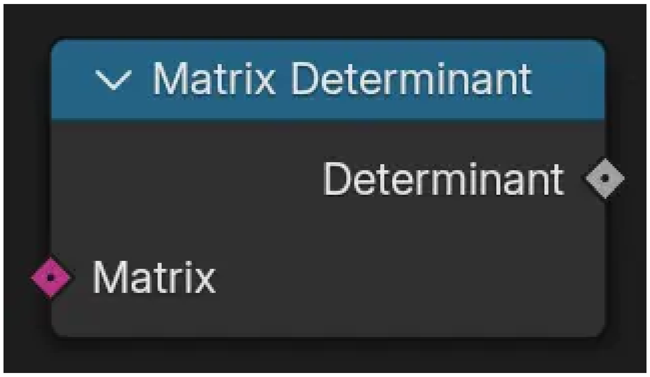
Matrix
: The inverted matrix.

Invertible
: Returns whether the matrix could be inverted. This can be false when a transformation matrix has a scale of zero, for example. See Invertible matrix for more information.

> **Important**
>
> When a matrix is not invertible, the identity matrix is returned.

Matrix Determinant Node

The *Matrix Determinant* node computes the determinant of the passed in matrix.


Matrix Determinant
Determinant
Matrix

Inputs

Matrix

The matrix to compute the determinant of.

Properties

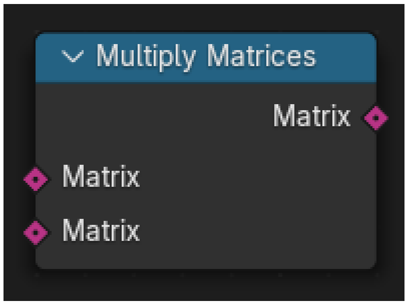
This node has no properties.

Outputs

Determinant

The compute determinant value.

Multiply Matrices Node


Multiply Matrices
Matrix
Matrix
Matrix

The *Multiply Matrices* node performs a matrix multiplication on two input matrices.

Inputs

Matrix
: The first multiplication.

Matrix
: The second multiplication.

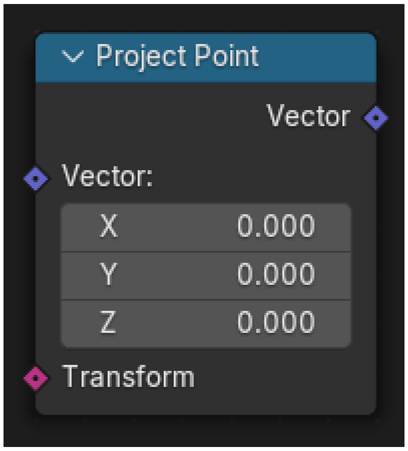
Properties

This node has no properties.

Outputs

Matrix
: The resulting matrix multiplication.

Project Point Node



Applies a projection matrix to a point. Specifically, this node turns the given Euclidean vector (X, Y, Z) into the homogeneous vector (X, Y, Z, 1), multiplies the given projection matrix by it, and turns the resulting homogeneous vector back into a Euclidean one by dividing it by the absolute value of its W component. This last step is also known as perspective division.

Inputs

Vector
: The position vector to project.

Transformation
: The projection matrix.

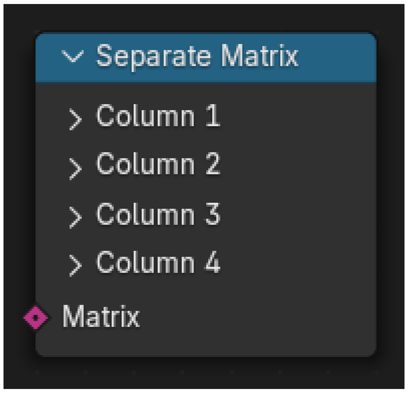
Properties

This node has no properties.

Outputs

Vector
: The projected position vector.

Separate Matrix Node


Separate Matrix
Column 1
Column 2
Column 3
Column 4
Matrix

The *Separate Matrix* node splits a 4x4 matrix into its individual values.

Inputs

Matrix
The matrix to split into individual values.

Properties

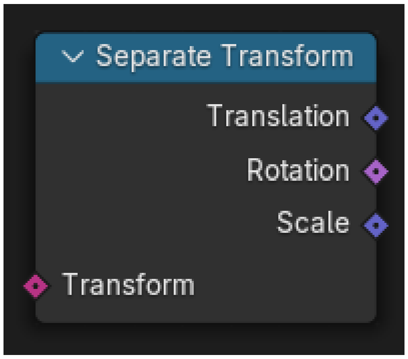
This node has no properties.

Outputs

The outputs of this node are split into panels for each column of the matrix. Each panel, has four value outputs for the four rows of the matrix.

Separate Transform Node

The *Separate Transform* node separates a Transformation Matrix into a translation vector, a rotation vector, and a scale vector.


Separate Transform
Translation
Rotation
Scale
Transform

Inputs

Transform
: The transformation matrix to separate.

Properties

This node has no properties.

Outputs

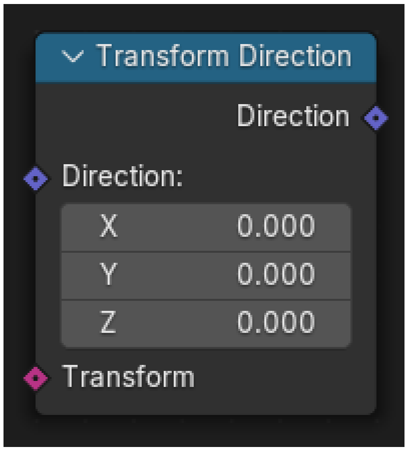
Translation
: The translation vector.

Rotation
: The rotation vector.

Scale
: The scale vector.

Transform Direction Node

The *Transform Direction* node multiplies a Transformation Matrix by a vector.


Transform Direction
Direction
Direction:
X 0.000
Y 0.000
Z 0.000
Transform

Inputs

Direction
: The vector.

Transformation
: The transformation matrix.

Properties

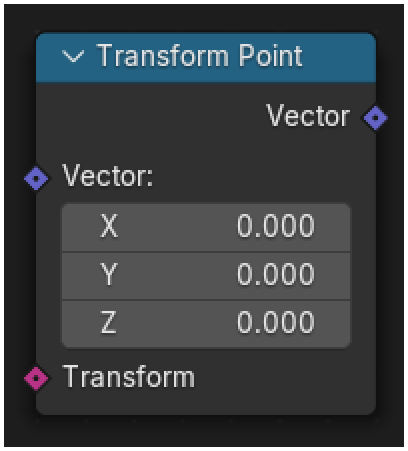
This node has no properties.

Outputs

Direction
: The vector.

Transform Point Node

The *Transform Point* node applies a Transformation Matrix to a position vector.


Transform Point
Vector
Vector:
X 0.000
Y 0.000
Z 0.000
Transform

Inputs

Vector
: The position of a point to transform.

Transformation
: The transformation matrix.

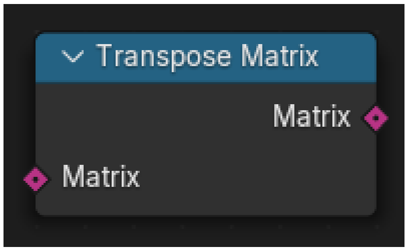
Properties

This node has no properties.

Outputs

Vector
: The transformed point.

Transpose Matrix Node


Transpose Matrix
Matrix
Matrix

The *Transpose Matrix* node flips a matrix over its diagonal.

See also

Transpose on Wikipedia.

Inputs

Matrix
: The matrix to be transposed.

Properties

This node has no properties.

Outputs

Matrix
: The transposed matrix.

Rotation Utility Nodes

- Align Rotation to Vector Node
- Axes to Rotation Node
- Axis Angle to Rotation Node
- Euler to Rotation Node
- Invert Rotation Node
- Rotate Rotation Node
- Rotate Vector Node
- Rotation to Euler Node
- Rotation to Quaternion Node
- Quaternion to Rotation Node

Align Rotation to Vector Node

Align Rotation to Vector node.

The *Align Rotation to Vector* node rotates an Euler rotation into the given direction.

Inputs

Rotation

The Euler rotation to align.

Important

This input has to be a rotation input. Be careful not to connect a direction vector like the normal.

Factor

Determines how much the points are rotated towards the vector. Zero effectively disables the node and one means that the points are aligned with the vector perfectly.

Vector

The direction vector that points should be rotated to. The vector is in the local space of the object that is being modified. When it is all zeros for a point, it is not rotated at all.

Properties

Axis

Local axis of the object that is to be rotated towards the vector input.

Pivot

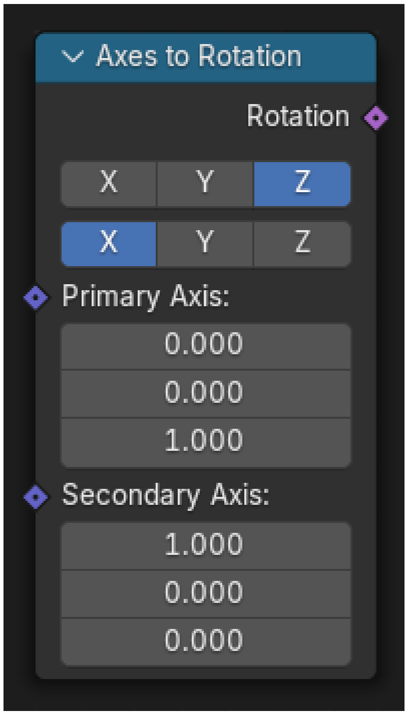
The local axis to rotate around.

Auto::	The best rotation angle is computed automatically. This minimizes the angle of rotation.
X, Y, Z::	Rotate around a specific local axis.

Outputs

Rotation
    The rotated Euler rotation.

Axes to Rotation Node


Axes to Rotation
Rotation
X
Y
Z
X
Y
Z
Primary Axis:
0.000
0.000
1.000
Secondary Axis:
1.000
0.000
0.000

Creates a rotation based on two axis directions.

Tip

In many cases, these directions are a normal and tangent on a mesh or curve.

Inputs

Primary Axis
The desired direction of the primary axis.

Secondary Axis
The desired direction of the secondary axis. Ideally, this is orthogonal to the primary direction.

Properties

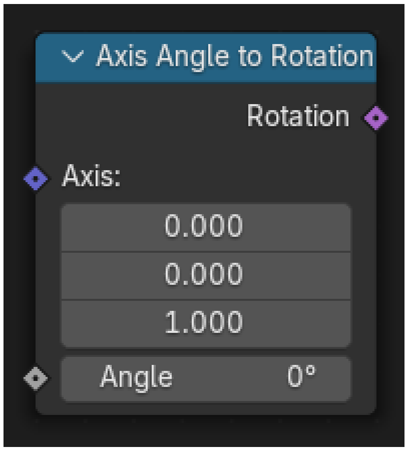
Primary Axis
The axis (X, Y or Z) that should be aligned exactly to the primary direction.

Secondary Axis
: The axis that should be aligned as closely as possible to the secondary direction.

Outputs

Rotation
: The rotation that results in the given axes being aligned to the given directions.

Axis Angle to Rotation Node


Axis Angle to Rotation
Rotation
Axis:
0.000
0.000
1.000
Angle
0°

The *Axis Angle to Rotation* node converts a axis angle rotation to a standard rotation value.

Inputs

Axis
: Unit vector representing the axis to rotate around.

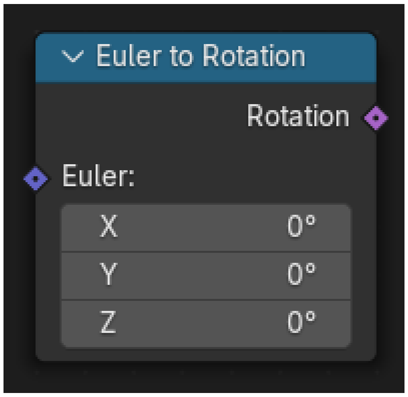
Angle
: The rotation angle around the axis.

Outputs

Rotation
: Standard rotation value.

Euler to Rotation Node

The *Euler to Rotation* node creates a rotation value from an Euler rotation.


Euler to Rotation
Rotation
Euler:
X 0°
Y 0°
Z 0°

Inputs

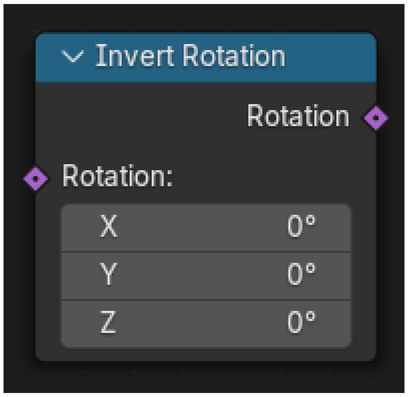
Euler
The Euler rotation.

Outputs

Rotation
Standard rotation value.

Invert Rotation Node

The *Invert Rotation* node inverts a rotation.


Invert Rotation
Rotation
Rotation:
X 0°
Y 0°
Z 0°

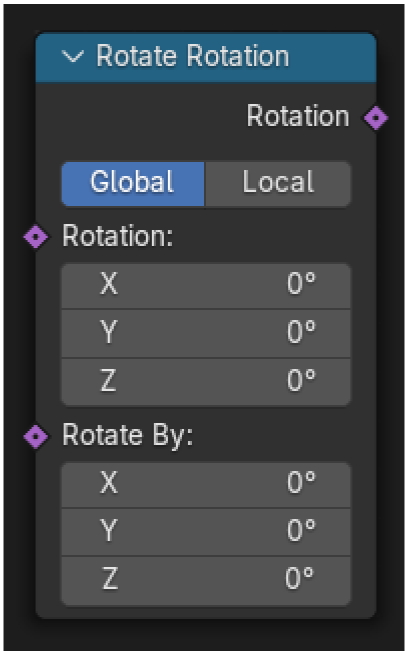
Inputs

Rotation
: Standard rotation value.

Outputs

Rotation
: The inverted rotation.

Rotate Rotation Node


Rotate Rotation
Rotation
Global
Local
Rotation:
X 0°
Y 0°
Z 0°
Rotate By:
X 0°
Y 0°
Z 0°

The *Rotate Rotation* node applies an additional rotation to a given one.

To rotate an Euler Rotation, first use the Euler to Rotation Node.

Inputs

Rotation
: The starting rotation.

Rotate By
: The additional rotation.

Properties

Space

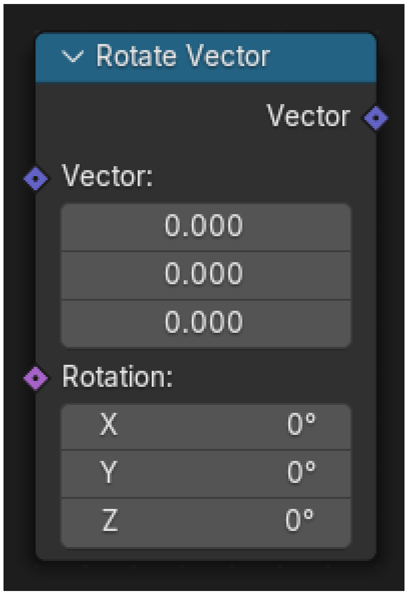
Global::	Rotate in Global Space.
Local::	Rotate in Local Space.

Outputs

Rotation
The resulting rotation.

Rotate Vector Node

The *Rotate Vector* node rotates a vector by a given rotation value.


Rotate Vector
Vector
Vector:
0.000
0.000
0.000
Rotation:
X 0°
Y 0°
Z 0°

Inputs

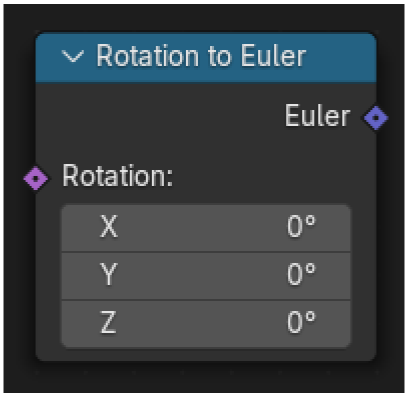
Vector
The vector to rotate.

Rotation
Standard rotation value.

Outputs

Vector
The rotated vector.

Rotation to Euler Node


Rotation to Euler
Euler
Rotation:

X 0°
Y 0°
Z 0°

The *Rotation to Euler* node converts a standard rotation socket value to an Euler rotation.

Inputs

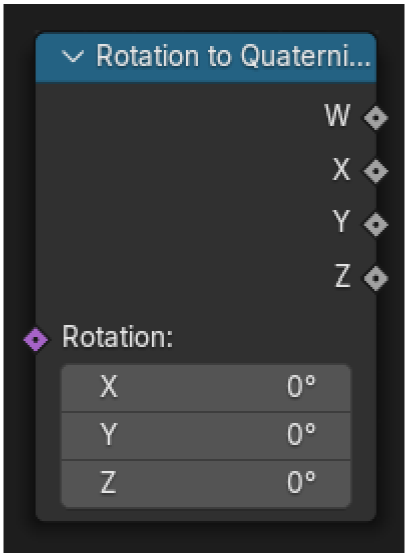
Rotation
: Standard rotation socket value.

Outputs

Euler
: The Euler rotation.

Rotation to Quaternion Node

The *Rotation to Quaternion* node converts a standard rotation value to a quaternion rotation.


Rotation to Quaterni...
W
X
Y
Z
Rotation:

X 0°
Y 0°
Z 0°

Inputs

Rotation

Standard rotation value.

Outputs

W

The W value of the quaternion.

X

The X value of the quaternion.

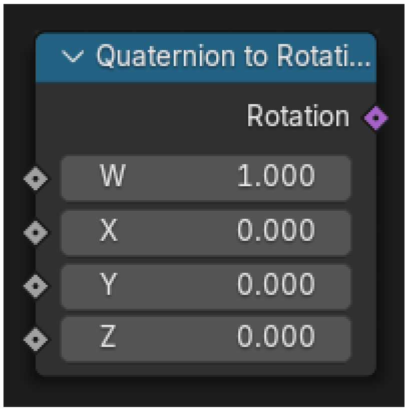
Y

The Y value of the quaternion.

Z

The Z value of the quaternion.

Quaternion to Rotation Node


Quaternion to Rotati...
Rotation
W
1.000
X
0.000
Y
0.000
Z
0.000

The *Quaternion to Rotation* node converts a quaternion rotation to a standard rotation.

Inputs

W
: The W value of the quaternion.

X
: The X value of the quaternion.

Y
: The Y value of the quaternion.

Z
: The Z value of the quaternion.

Outputs

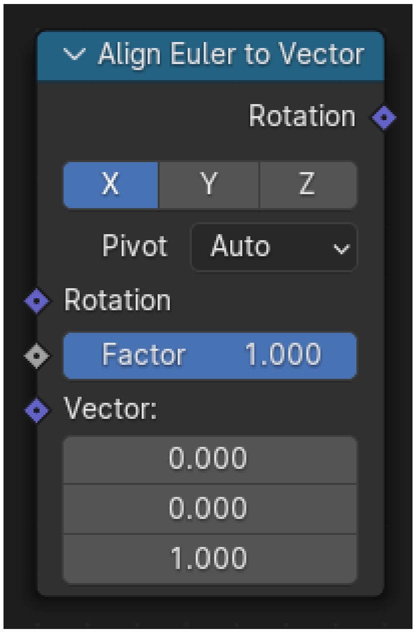
Rotation
: Standard rotation value.

Deprecated Nodes

These nodes have been made deprecated, meaning they will be removed in a future version and **should not be used**.

- Align Euler to Vector Node
- Rotate Euler Node

Align Euler to Vector Node


Align Euler to Vector
Rotation
X
Y
Z
Rotation
Vector:

Pivot Auto

Factor 1.000

0.000
0.000
1.000

The *Align Euler to Vector* node rotates an Euler rotation into the given direction.

Important

This node is deprecated, use the Align Rotation to Vector Node instead.

Inputs

Rotation
The Euler rotation to align.

Important

This input has to be a rotation input. Be careful not to connect a direction vector like the normal.

Factor

Determines how much the points are rotated towards the vector. Zero effectively disables the node and one means that the points are aligned with the vector perfectly.

Vector

The direction vector that points should be rotated to. The vector is in the local space of the object that is being modified. When it is all zeros for a point, it is not rotated at all.

Properties

Axis

Local axis of the object that is to be rotated towards the vector input.

Pivot

The local axis to rotate around.

Auto::	The best rotation angle is computed automatically. This minimizes the angle of rotation.
X, Y, Z::	Rotate around a specific local axis.

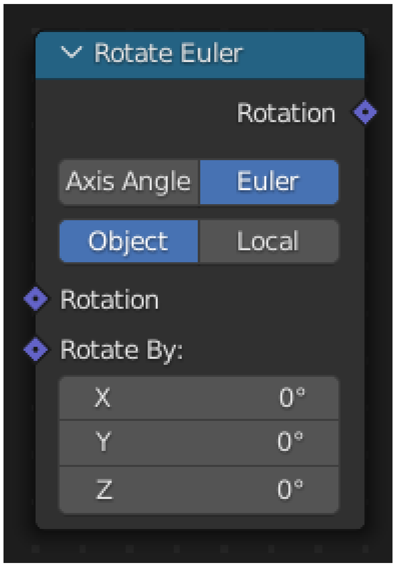
Outputs

Rotation

The rotated Euler rotation.

Rotate Euler Node

The *Rotate Euler* node rotates an Euler rotation.


Rotate Euler
Rotation
Axis Angle
Euler
Object
Local
Rotation
Rotate By:
X 0°
Y 0°
Z 0°

Important

This node is deprecated, use the Rotate Rotation Node instead.

Inputs

Rotation

The Euler rotation to rotate.

Rotate By

Specifies how much an Euler rotation is rotated. This input is only available when the rotation type is set to *Euler*.

Axis

The axis to rotate around. This input is only available when the rotation type is set to *Axis Angle*.

Angle

The angle to rotate by around the specified axis. This input is only available when the rotation type is set to *Axis Angle*.

Properties

Rotation Type

Axis Angle::	Use separate axis and angle inputs to control the rotation.
Euler::	Use an Euler input to control the rotation.

Space

Object::	Rotate an Euler rotation in the evaluated object's space.
Local::	Rotate an Euler rotation in local space.

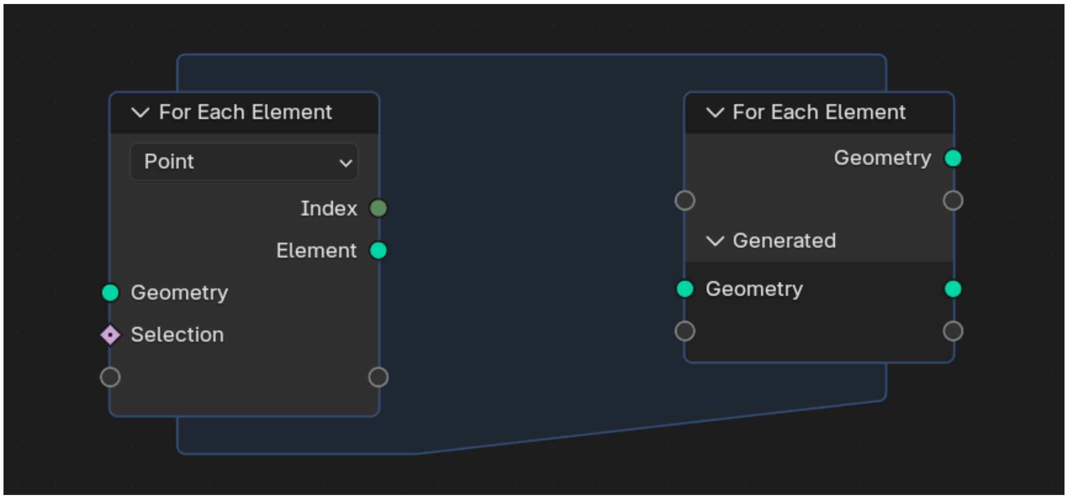
Outputs

Rotation

The rotated Euler rotation.

For Each Geometry Element Zone

This zone type allows executing nodes for each element of a geometry. For example, the nodes can process every face of a mesh, or every instance.



The *For Each Element* zone.

The zone is ideal for tasks that generate large or complex geometry for every element of an input geometry. For example, generating a unique tree for every input curve, or a unique building on every input face.

The zone makes less sense for processing small amounts of geometry. In that case (for example each of a character's hairs separately) it will likely **always be slower** than working on fewer larger geoemtries. The additional flexibility from processing each element separately comes at the cost that Blender can't optimize the operation as well. For node groups that need to handle lots of geometry elements, it's recommended to design the node setup so that iteration over tiny sub-geometries is not required.

Inputs

Geometry

Geometry whose elements are iterated over.

Selection

Which subset of the chosen *Domain* to process.

Index

Index of the element in the source geometry. Note that the same index can occure more than once when iterating over multiple geometry component types at once.

Element

The input geometry is split up into a separate geometry for each element. This is the single element geometry for the current iteration. This is not available for the *Face Corner* domain, since face corners cannot exist without their face.

Note

It can be quite inefficient to split up large geometries into many small elements. Because this output isn't computed if it's not used in the node graph, not using it will typically improve performance.

Properties

Domain

Which attribute domain to process.

Inspection Index

Geometry element index that is used by inspection features like the Viewer Node or socket inspection.

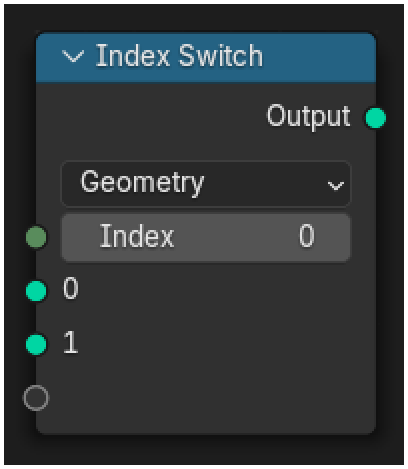
Outputs

The *Main Geometry* outputs create attributes on the "main" output geometry (the first output). Every single value on the inside of the zone becomes a value of the attribute at the current index.

The outputs in the *Generated* panel, including the default *Geometry* output are joined together from the geometry generated from each element. Any non-geometry type below a specific geometry in this list will output as an anonymous attribute on that joined geometry (and not the others). Attributes from the zone's input geometry are also propagated to these geometry outputs.

Index Switch Node

The *Index Switch* node outputs one of its inputs depending on an index value. Only the input that is passed through the node is computed.



See also

The [Menu Switch Node]() is similar but it exposes the choices as a menu.

Inputs

Index
: Determines which of the input options below will be passed through.

Item Inputs
: One input is created for every menu entry. The input is used when the matching option is selected.

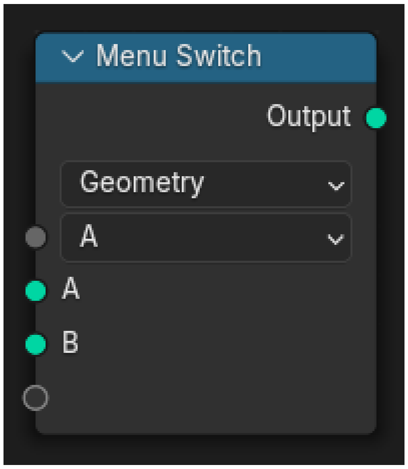
Properties

Type
: Determines the type of the data that is handled by the node.

Outputs

Output
: One of the inputs without any modifications.

Menu Switch Node

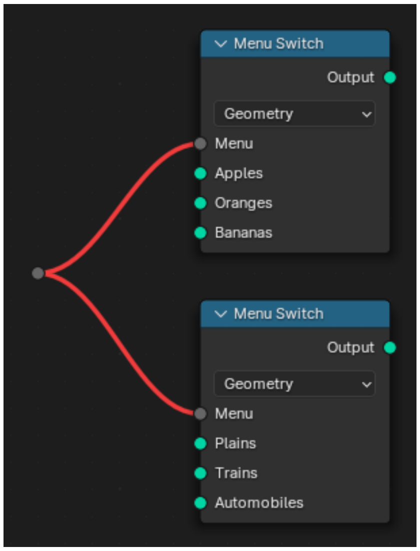


The *Menu Switch* node outputs one of its inputs depending on a menu selection. Only the input that is passed through the node is computed.

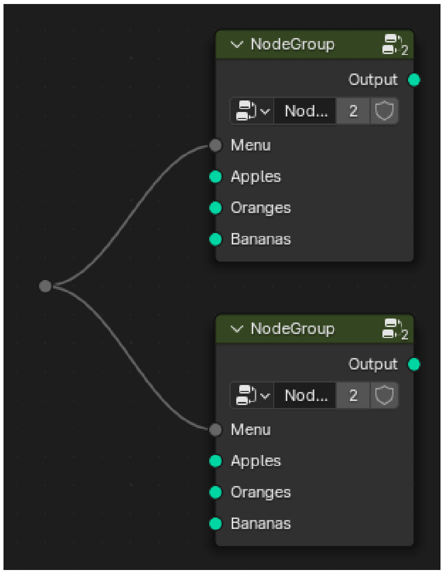
The available menu entries are defined by the user. Menu items can be added and removed, as well as renamed and reordered in the editor side bar. Renaming a menu entry keeps existing links of the matching input socket.

The menu can be used in node groups and the nodes modifier UI. Connecting the menu input with a *Group Input* node will expose the menu as a group input. A menu socket in a node group, reroute node, or other pass-through nodes needs to be connected to a *Menu Switch* node in order to work. An unconnected menu socket will show an empty menu by default.

Connecting multiple *Menu Switch* nodes to the same output socket creates a conflict (even when the menu entries are the same). To avoid this a menu switch can be wrapped in a node group. Multiple node groups of the same type can be connected to the same menu, since they contain the same menu switch node.


Menu Switch
Output
Geometry
Menu
Apples
Oranges
Bananas
Menu Switch
Output
Geometry
Menu
Plains
Trains
Automobiles

Conflict caused by connecting different menus.


NodeGroup
Output
Nod...
2
Menu
Apples
Oranges
Bananas
NodeGroup
Output
Nod...
2
Menu
Apples
Oranges
Bananas

Same node group can be connected without conflict.

See also

The Index Switch Node is similar but it exposes the choices as an integer index.

Inputs

Menu
: Determines which of the input options below will be passed through.

Item Inputs
: One input is created for every menu entry. The input is used when the matching option is selected. Items can be renamed by `Ctrl-LMB` on the socket name or in the nodes *Properties* panel.

Properties

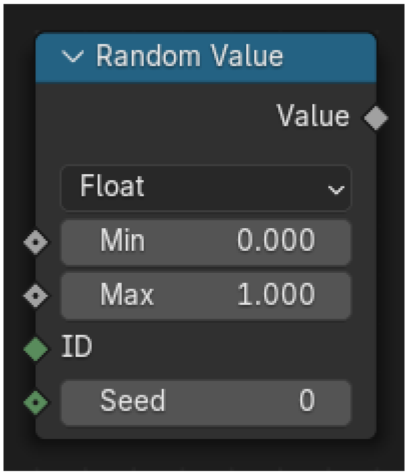
Type
    Determines the type of the data that is handled by the node.

Outputs

Output
    One of the inputs without any modifications.

Random Value Node

The *Random Value* node outputs a white noise like value as a *Float*, *Integer*, *Vector*, or *Boolean* field.



Inputs

Min

The minimum value of the range where random values are sampled from. This input is only available for *Float*, *Integer*, and *Vector* types.

Max

The maximum value of the range where random values are sampled from. This input is only available for *Float*, *Integer*, and *Vector* types.

Probability

The probability ratio for the random *Boolean* output to be *True*. This input is only available for *Boolean* types.

ID

An ID to drive the random number generator seed. By default, this input uses the same value as of the ID Node, which is the `id` attribute of the context geometry if it exists, and otherwise the index.

Tip

Single Random Value

By default, the random value node generates a value for each unique index. If a single random value is desired, connect a single value (such as an Integer Node) to the ID input.

Seed

A field to Seed the random number generator. This can be used to generate a different set of random values, even for two nodes with the same *ID* input.

Properties

Data Type

Float::	The output will be a *Float* field.
Integer::	The output will be an *Integer* field.
Vector::	The output will be a *Vector* field.
Boolean::	The output will be a *Boolean* field.

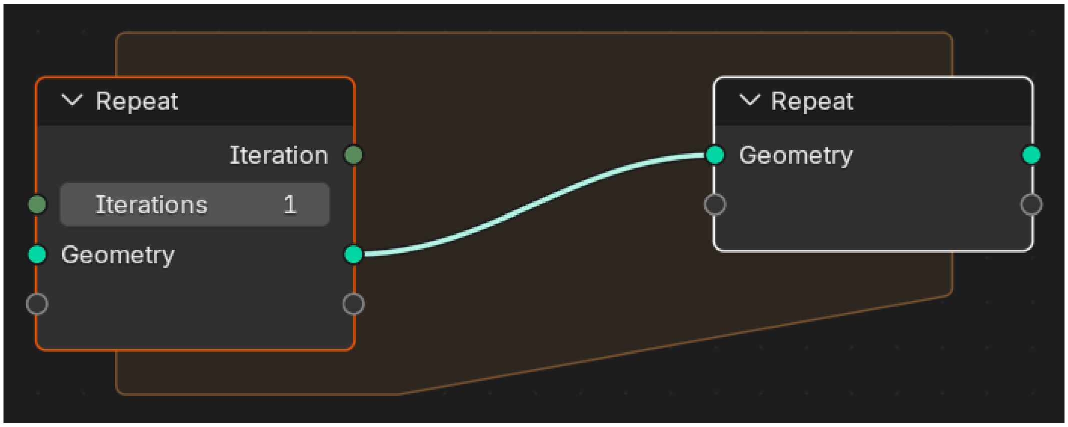
Outputs

Value

Random values as a field.

Repeat Zone

Repeat zones allow running nodes many times in a loop. Compared with simply duplicating a node, they support executing a node an arbitrary number of times, possibly determined when the node group is evaluated. For example, the nodes could be repeated based on the number of stories in a building generator.



Repeat zone used to repeat a node group a few times

When adding a repeat zone, two nodes are added, with the “zone” defined between them. The inputs connected to the *Repeat Input* node are read at the beginning, before starting the repetitions. Then they are passed to the inside of the zone where they can be changed, and passed to the next iteration. This process is repeated the specified number of times.

Other nodes can be connected as inputs to the inside of the repeat zone from the outside. Those are constant throughout every iteration based on their value at the current frame. However, outputs of the zone must be connected through the *Repeat Output* node.

Inputs

Iterations

Number of times to repeat the execution of the zone. The current iteration is available with the *Iteration* input on the inside of the zone.

Geometry

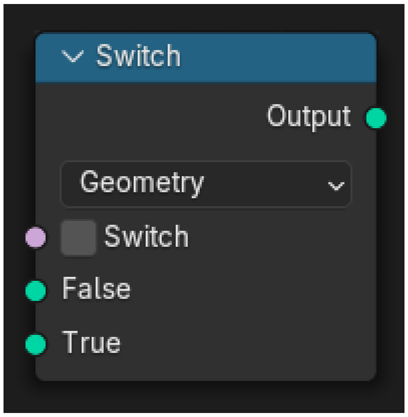
Standard geometry input, which is available by default to input geometry into the repeat zone. More bake items can be added by dragging sockets into the blank socket or in the *Bake Items* panel. Items can be renamed by `Ctrl-LMB` on the socket name or in the nodes *Properties* panel.

Properties

Inspection Index

Iteration number that is used by inspection features like the Viewer Node or socket inspection.

Switch Node



The *Switch* node outputs one of two inputs depending on a condition. Only the input that is passed through the node is computed.

See also

The Menu Switch Node and Index Switch Node can be used to switch between an arbitrary amount of inputs.

Inputs

Switch
: Determines which of the two inputs below will be passed through.

False
: Is passed through when the switch is set to false.

True
: Is passed through when the switch is set to true.

Properties

Type
: Determines the type of the data that is handled by the node.

Outputs

Output
: One of the two inputs without any modifications.

Group

A Group Node combines a set of nodes into a single one, and selectively exposes inputs and outputs of those nodes.

Group nodes can simplify a node tree by hiding away complexity and reusing functionality.

Group Input

Exposes the inputs of the node group. You can have multiple of these nodes in your tree to keep it clean, bringing in each input right where you need it (rather than dragging long links all across your graph).

The input slots can be edited in the *Group* tab of the *Sidebar*.

Group Output

Receives the outputs of the node group. You can have multiple of these nodes in your tree to keep it clean, outputting each result right where it's produced (rather than dragging long links all across your graph).

The output slots can be edited in the *Group* tab of the *Sidebar*.

Node Groups

This section lists all the node groups, both those in the current blend-file and those Linked or Appended from another blend-file.

Hair Nodes

Nodes focussed on generating or editing curves, typically used for hair.

- Deformation
    - Blend Hair Curves
    - Displace Hair Curves
    - Frizz Hair Curves
    - Hair Curves Noise
    - Roll Hair Curves
    - Rotate Hair Curves
    - Shrinkwrap Hair Curves
    - Smooth Hair Curves
    - Straighten Hair Curves
    - Trim Hair Curves
- Generation
    - Duplicate Hair Curves
    - Generate Hair Curves
    - Interpolate Hair Curves
- Guides
    - Braid Hair Curves
    - Clump Hair Curves
    - Create Guide Index Map
    - Curl Hair Curves
- Read
    - Curve Info
    - Curve Root
    - Curve Segment
    - Curve Tip
    - Hair Attachment Info
- Utility
    - Attach Hair Curves to Surface
    - Redistribute Curve Points
    - Restore Curve Segment Length
- Write

    - Set Hair Curve Profile

Hair Deformation Nodes

- Blend Hair Curves
- Displace Hair Curves
- Frizz Hair Curves
- Hair Curves Noise
- Roll Hair Curves
- Rotate Hair Curves
- Shrinkwrap Hair Curves
- Smooth Hair Curves
- Straighten Hair Curves
- Trim Hair Curves

Blend Hair Curves

Blends shape between multiple hair curves in a certain radius.

Inputs

Geometry
Input Geometry (only curves will be affected).

Factor
Factor to blend overall effect.

Blend Radius
Radius to select neighbors for blending.

Blend Neighbors
Amount of neighbors used for blending.

Preserve Length
: Preserve each curve’s length during deformation.

Properties

This node has no properties.

Outputs

Geometry

Displace Hair Curves

Displaces hair curves by a vector based on various options.

Inputs

Geometry
    Input Geometry (only curves will be affected).

Factor
    Factor to scale overall displacement.

Shape
    Shape of the influence along curves (0=constant, 0.5=linear).

Object Space
    Object used to define the displacement space.

Displace Vector
    Vector for displacement.

Surface
    Surface geometry used to sample the normal for displacement. This input takes priority over the corresponding object input, if used.

Surface
    Surface object used to sample the normal for displacement.

Surface UV Map
    Surface UV map used to sample the normal for displacement.

Surface Normal Displacement
    Amount of displacement along the surface normal.

Properties

This node has no properties.

Outputs

Geometry

Frizz Hair Curves

Deforms hair curves using a random vector per point to frizz them.

Inputs

Geometry
: Input Geometry (only curves will be affected).

Cumulative Offset
: Apply offset cumulatively (previous points affect points after).

Factor
: Factor to blend overall effect.

Distance
: Overall distance factor for the deformation.

Shape
    Shape of the influence along curves (0=constant, 0.5=linear).

Seed
    Random Seed for the operation.

Preserve Length
    Preserve each curve's length during deformation.

Properties

This node has no properties.

Outputs

Geometry

Offset Vector
    Vector by which each point was offset during deformation.

Hair Curves Noise

Deforms hair curves using a noise texture.

Inputs

Geometry

Cumulative Offset
  Apply offset cumulatively (previous points affect points after).

Factor
  Overall factor for the deformation.

Distance
  Overall distance factor for the deformation.

Shape

Shape of amount along each curve (0=constant, 0.5=linear).

Scale
Scale of the noise texture by root position.

Scale along Curve
Scale of noise texture along each Curve.

Offset per Curve
Random offset of noise texture for each Curve.

Seed
Seed value for randomization.

Preserve Length
Preserve the length of the Curves on a segment basis.

Properties

This node has no properties.

Outputs

Geometry

Offset Vector

Roll Hair Curves

Rolls up hair curves starting from their tips.

Inputs

Geometry

Factor
  Factor to blend overall effect.

Subdivision
  Subdivision level applied before deformation.

Variation Level
  Level of smoothing on the roll path to include shape variation.

Roll Length

Length of each curve to be rolled.

Roll Radius
Radius of the rolls.

Roll Depth
Depth offset of the roll.

Roll Taper
Taper of the roll.

Retain Overall Shape
Offset the roll along the original curve to retain shape.

Roll Direction
Axis around which each curve is rolled.

Random Orientation
Amount of randomization of the direction of the roll.

Seed
Random Seed for the operation.

Preserve Length
Preserve each curve’s length during deformation.

Properties

This node has no properties.

Outputs

Geometry

Rotate Hair Curves

Rotates each hair curve around an axis.

Inputs

Geometry
Input Geometry (only curves will be affected).

Factor
Factor to influence the rotation angle.

Axis
Rotation Axis (Default: Tangent at root).

Angle
Angle of rotation.

Random Offset
  Random offset to the rotation angle per Curve.

Lock Ends
  Lock rotation to the axis between the curve ends.

Seed
  Random Seed for the operation.

Properties

This node has no properties.

Outputs

Geometry

Shrinkwrap Hair Curves

Shrinkwraps hair curves to a mesh surface from below and optionally from above.

Inputs

Geometry
Input Geometry (only curves will be affected).

Surface
Surface geometry used for shrinkwrap. This input takes priority over the corresponding object input, if used.

Surface
Surface object used for shrinkwrap.

Factor

Offset Distance
Distance from the surface used for shrinkwrap.

Above Surface
Blend shrinkwrap for points above the surface.

Smoothing Steps
Amount of steps of smoothing applied after shrinkwrap.

Lock Roots
Lock the position of root points.

Properties

This node has no properties.

Outputs

Geometry

Smooth Hair Curves

Smooths the shape of hair curves.

Inputs

Geometry
    Input Geometry (only curves will be affected).

Amount
    Amount of smoothing. Negative values will result in crumpling the curves.

Shape
    Shape of the influence along curves (0=constant, 0.5=linear).

Iterations
    Amount of smoothing steps.

Weight
    Weight used for smoothing.

Lock Tips
    Lock tip position when smoothing.

Preserve Length
    Preserve each curve’s length during deformation.

Properties

This node has no properties.

Outputs

Geometry

Straighten Hair Curves

Straightens hair curves between root and tip.

Inputs

Geometry
: Input Geometry (only curves will be affected).

Amount
: Amount of straightening. Negative values will result in crumpling the curves.

Shape
: Shape of the influence along curves (0=constant, 0.5=linear).

Preserve Length
: Preserve each curve’s length during deformation.

Properties

This node has no properties.

Outputs

Geometry

Trim Hair Curves

Trims or scales hair curves to a certain length.

Inputs

Geometry
Input Geometry (only curves will be affected).

Scale Uniform
Scale each curve uniformly to reach the target length.

Length Factor
Multiply the original length by a factor.

Replace Length
Use the length input to fully replace the original length.

Length
: Target length for the operation.

Mask
: Mask to blend overall effect.

Random Offset
: Trim hair curves randomly up to a certain amount.

Pin at Parameter
: Pin each curve at a certain point for the operation.

Seed
: Random Seed for the operation.

Properties

This node has no properties.

Outputs

Geometry

Hair Generation Nodes

- Duplicate Hair Curves
- Generate Hair Curves
- Interpolate Hair Curves

Duplicate Hair Curves

Duplicates hair curves a certain number of times within a radius.

Inputs

Geometry
: Input Geometry (only curves will be affected).

Amount
: Amount of duplicates per curve.

Viewport Amount
: Percentage of amount used for the viewport.

Radius
: Radius in which the duplicate curves are offset from the guides.

Distribution Shape
Shape of distribution from center to the edge around the guide.

Tip Roundness
Offset of the curves to round the tip.

Even Thickness
Keep an even thickness of the distribution of duplicates.

Seed
Random Seed for the operation.

Properties

This node has no properties.

Outputs

Geometry

Guide Index
Guide index map that was used for the operation.

Generate Hair Curves

Generates new hair curves on a surface mesh. The curves are generated from scratch at point locations; if creating curves that depend on existing curves is desired, the Interpolate Hair Curves is a better choice.

Note

This node/modifier will not function without the *Surface* geometry/object and *Surface UV Map* inputs.

Inputs

Surface

Surface geometry for generation. This input takes priority over the corresponding object input if both are provided.

Surface

Surface object for generation (The transforms of this object must match the modifier object).

Surface UV Map

Surface UV map stored on the mesh used for finding curve attachment locations.

Surface Rest Position

Set the surface mesh into its rest position before attachment.

Tip

In a typical hair generation setup, this node or modifier will be combined with the Deform Curves on Surface Node. If that operation comes after this one, it makes sense to turn this option on so the position used is the pre-deformed position consistent with the expectations for the deformation's input.

Hair Length

Length of the generated hair curves.

Hair Material

Material of the generated hair curves.

Control Points

Amount of control points of the generated hair curves.

Poisson Disk Distribution

Use poisson disk distribution method to keep a minimum distance. See the Distribute Points on Faces for more information.

Density

Surface density of generated hair curves.

Density Mask

Factor applied on the density for curve distribution.

Mask Texture

Discard points based on an mask texture after distribution. The image is sampled with the *Surface UV Map* input.

Tip

The accuracy of sampling the image doesn't depend on the density of the surface mesh's vertices because it is sampled after the curve root points are generated, the accuracy . However, using the *Density Mask* input instead can give better performance. Using them in combination can give the benefits of both methods.

Viewport Amount

Factor applied on the density for the viewport.

Seed

Random seed for the operation.

Properties

This node has no properties.

Outputs

Geometry

Curves

Surface Normal

Normal direction of the surface mesh at the attachment point.

Interpolate Hair Curves

Interpolates existing guide curves on a surface mesh. The Duplicate Hair Curves is a similar option with simpler behavior that may offer better performance.

Note

This node/modifier will not function without the *Surface* geometry/object and *Surface UV Map* inputs.

Inputs

Geometry

Input Geometry (only curves will be affected).

Surface

Surface geometry for generation. This input takes priority over the corresponding object input if both are provided.

Surface

Surface object for generation (Needs matching transforms).

Surface UV Map

Surface UV map stored on the mesh used for finding curve attachment locations.

Surface Rest Position

Set the surface mesh into its rest position before attachment.

Tip

In a typical hair generation setup, this node or modifier will be combined with the Deform Curves on Surface Node. If that operation comes after this one, it makes sense to turn this option on so the position used is the pre-deformed position consistent with the expectations for the deformation's input.

Follow Surface Normal

Align the interpolated curves to the surface normal.

Part by Mesh Islands

Use mesh islands of the surface geometry for parting.

Interpolation Guides

Amount of guides to be used for interpolation per curve.

Distance to Guides

Distance around each guide to spawn interpolated curves.

Poisson Disk Distribution

Use poisson disk distribution method to keep a minimum distance.

Density

Surface density of generated hair curves.

Density Mask
Factor applied on the density for curve distribution.

Mask Texture
Discard points based on an mask texture after distribution. The image is sampled with the *Surface UV Map* input.

> **Tip**
>
> The accuracy of sampling the image doesn't depend on the density of the surface mesh's vertices because it is sampled after the curve root points are generated, the accuracy . However, using the *Density Mask* input instead can give better performance. Using them in combination can give the benefits of both methods.

Viewport Amount
Factor applied on the density for the viewport.

Seed
Random seed for the operation.

Properties

This node has no properties.

Outputs

Geometry

Guide Index
Index of the main guide curve per curve.

Surface Normal
Normal direction of the surface mesh at the attachment point.

Hair Guides Nodes

- Braid Hair Curves
- Clump Hair Curves
- Create Guide Index Map
- Curl Hair Curves

Braid Hair Curves

Deforms existing hair curves into braids using guide curves.

Inputs

Geometry

Guide Index

Guide index map witch describes which curve to use as the center of each braid group. If this input is provided, it priority over an existing map in the `guide_curve_index` attribute, and the *Guide Distance* and *Guide Mask* attribute will be unused.

Guide Distance

Minimum distance between two guides for new guide map.

Guide Mask

Mask for which curve are eligible to be selected as guides.

Existing Guide Map
Use the existing guide map attribute if available. If this is false, and the *Guide Index* input isn’t provided, the *Guide Distance* and *Guide Mask* input will be used to generate a new guide map for this node. Creating the guide map in a separate node or modifier gives more complete control over its creation.

Factor
Factor to blend overall effect.

Subdivision
Subdivision level applied before deformation.

Braid Start
Percentage along each curve to blend deformation from the root.

Radius
Overall radius of the braids.

Shape
Shape of the braid radius along each curve.

Factor Min
Factor of the minimum radius of the braids.

Factor Max
Factor of the maximum radius of the braids.

Frequency
Frequency factor of the braids. This input can vary for different points of the same curve.

Thickness
Thickness of each strand of hair.

Thickness Shape
Shape adjustment of the strand thickness for the braids.

Shape Asymmetry
Asymmetry of the shape adjustment of the strand thickness.

Flare Length
Length of the flare at the end of the braid.

Flare Opening
Opening radius of the flare at the tip of the braid.

Hair Tie
Geometry used for the hair tie instance (priority).

Hair Tie
Object used for the hair tie instance.

Hair Tie Scale
Scale of the hair tie instance.

Properties

This node has no properties.

Outputs

Geometry

Guide Index
Guide index map that was used for the operation. If a new guide map is created by this node, it will be stored for this output.

Flare Parameter
Parameter from 0 to 1 along the flare.

Strand Index
Index of the group of hair in the braid that each hair curve belongs to.

Clump Hair Curves

Clumps together existing hair curves using guide curves.

Inputs

Geometry

Input Geometry (only curves will be affected).

Guide Index

Guide index map witch describes which curve to use as the center of each braid group. If this input is provided, it priority over an existing map in the `guide_curve_index` attribute, and the *Guide Distance* and *Guide Mask* attribute will be unused.

Guide Distance

Minimum distance between two guides for new guide map.

Guide Mask
: Mask for which curve are eligible to be selected as guides.

Existing Guide Map
: Use the existing guide map attribute if available. If this is false, and the *Guide Index* input isn't provided, the *Guide Distance* and *Guide Mask* input will be used to generate a new guide map for this node. Creating the guide map in a separate node or modifier gives more complete control over its creation.

Factor
: Factor to blend overall effect.

Shape
: Shape of the influence along curves (0=constant, 0.5=linear).

Tip Spread
: Distance of random spread at the curve tips.

Clump Offset
: Offset of each clump in a random direction.

Distance Falloff
: Falloff distance for the clumping effect (0 means no falloff).

Distance Threshold
: Distance threshold for the falloff around the guide.

Seed
: Random seed for the operation.

Preserve Length
: Preserve each curve's length during deformation.

Properties

This node has no properties.

Outputs

Geometry

Guide Index

Guide index map that was used for the operation. If a new guide map is created by this node, it will be stored for this output.

Create Guide Index Map

Creates an integer attribute named `guide_curve_index` that stores the nearest guide curve for every curve to its nearest guide via index.

Other nodes in the Hair Guides Nodes category can generate guide maps themselves for convenience, but the behavior is always the same as this node.

Inputs

Geometry

Guides

Guide Curves or Points used for the selection of Guide Curves.

Guide Distance

Minimum distance between two guides.

Guide Mask
: Mask for which curve are eligible to be selected as guides.

Group ID
: ID to group together curves for guide map creation. Curves will only choose a guide with the same ID value.

Properties

This node has no properties.

Outputs

Geometry
: Output geometry including the new map attribute and the guide selection anonymous attribute as well. This geometry includes the guide curves, they are not separated.

Guide Curves
: Output geometry including only the selected guide curves.

Guide Index
: The index of the closest curve with the same *Group ID* value.

Guide Selection
: A selection in the *Geometry* output set to true for only the curves that were chosen as guides.

Curl Hair Curves

Deforms existing hair curves into curls using guide curves.

Inputs

Geometry

Guide Index

Guide index map witch describes which curve to use as the center of each braid group. If this input is provided, it priority over an existing map in the `guide_curve_index` attribute, and the *Guide Distance* and *Guide Mask* attribute will be unused.

Guide Distance

Minimum distance between two guides for new guide map.

Guide Mask

Mask for which curve are eligible to be selected as guides.

Existing Guide Map

Use the existing guide map attribute if available. If this is false, and the *Guide Index* input isn't provided, the *Guide Distance* and *Guide Mask* input will be used to generate a new guide map for this node. Creating the guide map in a separate node or modifier gives more complete control over its creation.

Factor

Factor to blend overall effect.

Subdivision

Subdivision level applied before deformation.

Curl Start

Percentage along each curve to blend deformation from the root.

Radius

Overall radius of the curls.

Factor Start

Factor for the radius at the curl start.

Factor End

Factor for the radius at the curl end.

Frequency

Frequency factor of the curls. This input can vary for different points of the same curve.

Random Offset

Amount of random offset per curve.

Seed

Random Seed for the operation.

Properties

This node has no properties.

Outputs

Geometry

Guide Index
: Guide index map that was used for the operation. If a new guide map is created by this node, it will be stored for this output.

Hair Read Nodes

- Curve Info
- Curve Root
- Curve Segment
- Curve Tip
- Hair Attachment Info

Curve Info

Reads information about each curve.

Inputs

This node has no inputs.

Properties

This node has no properties.

Outputs

Curve Index
    Index of each Curve.

Curve ID
ID of each Curve.

Length
Length of each Curve.

Direction
Direction from root to tip of each Curve.

Random
Random vector for each Curve.

Surface UV
Attachment surface UV coordinate of each Curve.

Curve Root

Reads information about each curve's root point.

Inputs

This node has no inputs.

Properties

This node has no properties.

Outputs

Root Selection

Boolean selection of curve root points.

Root Position	Position of the root point of a Curve.
Root Direction	Direction of the root segment of a Curve.
Root Index	Index of the root point of a Curve.

Curve Segment

Reads information each point's previous curve segment.

Inputs

This node has no inputs.

Properties

This node has no properties.

Outputs

Segment Length
    Distance to previous point on Curve.

Segment Direction
    Direction from previous neighboring point on segment.

Neighbor Index
    Index of previous neighboring point on segment.

Curve Tip

Reads information about each curve’s tip point.

Inputs

This node has no inputs.

Properties

This node has no properties.

Outputs

Tip Selection
Boolean selection of curve tip points.

Tip Position	Position of the tip point of a Curve.
Tip Direction	Direction of the tip segment of a Curve.
Tip Index	Index of the tip point of a Curve.

Hair Attachment Info

Reads attachment information regarding a surface mesh.

Inputs

Surface Geometry
: Surface geometry of the curve attachment.

Surface UV Map
: Surface UV map stored on the mesh used for finding curve attachment locations.

Properties

This node has no properties.

Outputs

Attachment UV
: Surface attachment UV coordinate stored on each curve.

Attachment is Valid
: Whether the stored attachment UV coordinate is valid.

Surface Normal
: Normal direction of the surface mesh at the attachment point.

Hair Utility Nodes

- Attach Hair Curves to Surface
- Redistribute Curve Points
- Restore Curve Segment Length

Attach Hair Curves to Surface

Attaches hair curves to a surface mesh.

Note

This node/modifier will not function without the *Surface* geometry/object and *Surface UV Map* inputs.

Inputs

Geometry

Input Geometry (only curves will be affected).

Surface

Surface Geometry to attach hair curves to. This input takes priority over the corresponding object input, if used.

Surface

Surface Object to attach to (needs to have matching transforms).

Surface UV Map

Surface UV map stored on the mesh used for finding curve attachment locations.

Surface Rest Position

Set the surface mesh into its rest position before attachment.

> **Tip**
>
> In a typical hair generation setup, this node or modifier will be combined with the Deform Curves on Surface Node. If that operation comes after this one, it makes sense to turn this option on so the position used is the pre-deformed position consistent with the expectations for the deformation's input.

Sample Attachment UV

Sample the surface UV map at the attachment point.

Snap to Surface

Snap the root of each curve to the closest surface point.

Align to Surface Normal

Align the curve to the surface normal (need guide as reference).

Blend along Curve

Blend deformation along each curve from the root.

Properties

This node has no properties.

Outputs

Geometry

Surface UV Coordinate
    Surface UV coordinate at the attachment point.

Surface Normal
    Surface normal at the attachment point.

Redistribute Curve Points

Redistributes existing control points evenly along each curve.

Inputs

Curves

Factor
    Factor to blend overall effect.

Feature Awareness
    Use simple feature awareness to keep feature definition.

Properties

This node has no properties.

Outputs

Curves

Restore Curve Segment Length

Restores the length of each curve segment using a previous state after deformation.

Inputs

Curves

Selection
Only affect selected elements.

Factor
Factor to blend overall effect.

Reference Position
Reference position before deformation.

Pin at Parameter

Pin each curve at a certain point for the operation.

Properties

This node has no properties.

Outputs

Curves

Write Hair Nodes

- Set Hair Curve Profile

Set Hair Curve Profile

Sets the radius attribute of hair curves according to a profile shape.

Inputs

Geometry

Replace Radius
: Replace the original radius.

Radius
: Base radius to be set if *Replace Radius* is enabled.

Shape
: Shape of the radius along the curve.

Factor Min

Factor of the radius at the minimum.

Factor Max
    Factor of the radius at the maximum.

Properties

This node has no properties.

Outputs

Geometry

Normals Nodes

- Smooth By Angle Node Group

Smooth By Angle Node Group

Set the sharpness of mesh edges based on the angle between the neighboring faces.

Note

This is a node group asset that is included in the bundled “Essentials” asset library.

Inputs

Mesh
: Standard geometry input.

Angle
: Maximum angle between face normals that will be considered as smooth.

Ignore Sharpness
: Smooth all edges, even if they have been marked as sharp.

Properties

This node has no properties.

Outputs

Mesh
: Standard geometry output.

Transform

- Introduction
  - Operators
- Transform Modal Map
  - Constraints
  - Snapping
  - Navigating

Introduction

Transform is the modality of operations that perform transformations in 2D and 3D elements. Transformations can include things like moving, rotating, scaling, and applying other operations to objects in the scene.

They work by changing the geometry which you can edit directly.

Operators

There are several transformation operations included in Blender. Here are some of the main operations available:

Move

This operations allows you to move elements along the X, Y, and Z axes in the scene.

Rotate

You can use this function to rotate elements around the X, Y, and Z axes.

Scale

Scaling allows you to increase or decrease the size of an object along the X, Y, and Z axes.

Align to View

This is useful for aligning objects with the view from the camera or another specific viewpoint.

Mirror

Mirrors objects along one or more axes.

Transform Modal Map

During a transformation, some hotkeys can change the behavior of the operation.

You can check editing the keys of these modal modifiers in *Blender Preferences* ▸ *Keymap* ▸ *Transform Modal Map* (at the bottom of the keymap).

Constraints

When moving, rotating or scaling, if you only want certain axes to be affected, you can restrict the transformation to those axes.

By default the constraint keys are `X`, `Y` and `Z`. This constraint can be restricted to a plane if `Shift` is pressed or automatically detected if `MMB` is pressed.

It is worth noting that if you press the same contraint hotkey a second time, you change the orientation from Local to Global or vice versa. Pressing a third time disables the constraint.

Snapping

Transform operations use the snapping settings set in the scene. However, some options can be changed during the transformation.

Snap Invert

Even if the magnetic icon is disabled, you can still enable snapping during a transformation. The default hotkey in this case is `Ctrl`.

Set Snap Base

The Snap Base is automatically determined based on the Snap Base options. However, this automatic snap detection point of origin may not always align with the user's intentions. Therefore, transform operations include a utility to set a new snap origin point during the transformation. The new Snap Base will correspond to the snap point whose target is defined by the Snap Target.

By default the hotkey is `B`.

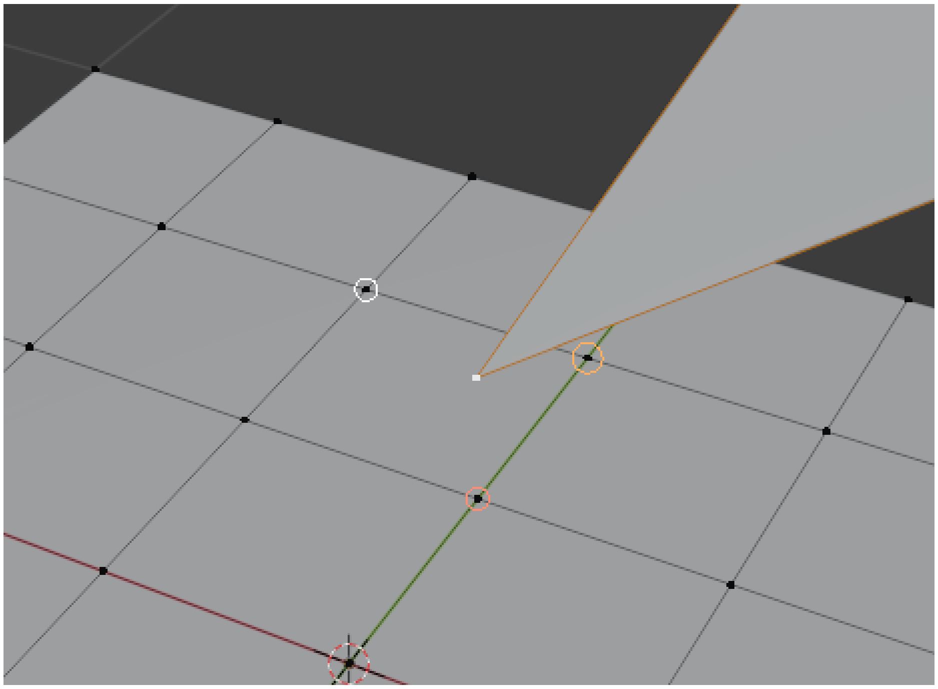
Note

If only Snap to Increment is enabled as Snap Target, the targets **Vertex**, **Edge**, **Face** and **Edge Center** will be used instead.

Add Snap Point

While you're transforming a selection with snapping enabled, you can press `A` whenever there's a highlighted snap target to mark it. With multiple such targets marked, the selection will then be snapped to their average location.

Marking a target more than once will give it more weight.



Multiple snapping targets.

Navigating

While performing a modal transformation, you can perform navigation actions such as zooming, panning, or rotating by holding `Alt` then perform the desired action.

Sculpting & Painting

- Introduction
- Brushes
    - Introduction
    - Manage Brushes
    - Brush Settings
- Selection & Visibility
    - Selection Masking
    - Select Linked
- Navigation

Modes

- Sculpting
    - Introduction
    - Brushes
    - Toolbar
    - Tools
    - Tool Settings
    - Controls
    - Editing
- Texture Paint
    - Introduction
    - Brushes
    - Tool Settings
- Vertex Paint
    - Introduction
    - Brushes
    - Vertex Paint Tools
    - Tool Settings
    - Editing
- Weight Paint
    - Introduction

  - Brushes
  - Weight Paint Tools
  - Tool Settings
  - Using Vertex Groups
  - Editing
- Curves Sculpting
  - Introduction
  - Brushes
  - Common Settings

Introduction

Sculpting and painting offers a more freeform workflow of editing via brushes. There are several modes to do this, each with their own purpose.

- Sculpting: Change and transform the topology of your mesh.
- Vertex Paint: Change the color of vertices in the active Color Attribute.
- Weight Paint: Change the weight of vertices in the active vertex group.
- Texture Paint: Change the pixels of the active image texture.

Brushes

- Introduction
    - Accessing Brushes
    - Brush Control
    - Custom Brush Shortcuts
    - Brush Assets
    - Brush Tool
- Manage Brushes
    - Asset Operators
    - Manual Storage

Brush Settings

- Brush Settings
- Texture & Texture Mask
- Stroke
- Falloff
- Cursor

Introduction

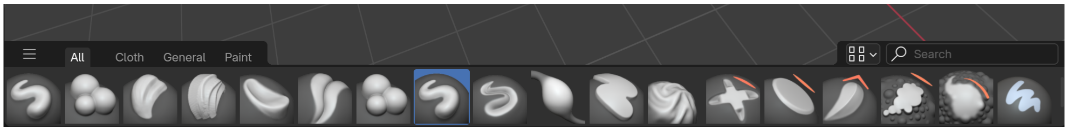
Brushes are the main way of interacting with any painting and sculpting mode. By click & dragging in the 3D Viewport (or the Image Editor when using Texture Paint), the active brush creates a stroke with a certain effect, depending on the used brush settings. Brushes are used as brush assets and stored in asset libraries, which makes it easy to reuse and share them. Typically they have a preview image and a name that indicate the effect they create.

Tip

It is highly recommended to use a Graphics Tablet for a better brush feel and additional features.

Accessing Brushes

In modes that use painting or sculpting functionality, the Asset Shelf of the 3D Viewport and Image Editor displays brush assets that can be used in that mode. Clicking a brush asset will activate the Brush Tool if necessary, with the clicked brush set.



The Asset Shelf of the 3D Viewport, providing access to brush assets.

This asset shelf is also available as popup in the Tool Settings, the Sidebar, Properties and using a shortcut.

Reference

Mode::	All Paint Modes
Header::	*Tool Settings*
Panel::	*Sidebar* ▸ *Tool* ▸ *Brush Asset*, *Properties* ▸ *Tool* ▸ *Brush Asset*
Shortcut::	`Shift-Spacebar`

Brush Control

These are the most common hotkeys for controlling the brush.

- Set brush size `F`
- Set brush strength `Shift-F`
- Rotate brush texture / Set brush weight `Ctrl-F`

After pressing these hotkeys, you can then either adjust the value interactively or by typing in numbers. Move the mouse right or left to increase/reduce the value (additionally with precision (`Shift`) and/or snapping (`Ctrl`) activated). Finally confirm (`LMB`, `Return`) or cancel (`RMB`, `Esc`).

You can also invert the brush direction/effect by holding `Ctrl`.

Custom Brush Shortcuts

To give a brush a shortcut, simply right click it in the asset shelf or brush selector popup, and select *Assign Shortcut*. To modify or remove an existing shortcut, select *Change Shortcut* or *Remove Shortcut* accordingly.

Brush Assets

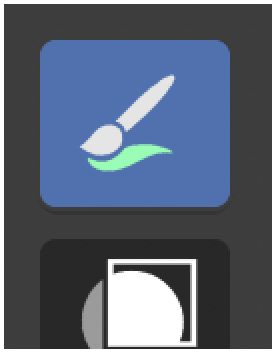
Brushes are used as assets, and stored in asset libraries. This makes the brushes shared across project files. All available brush assets can be displayed in the Asset Browser, which also provides ways to organize them.

Blender comes bundled with a number of brushes in the *Essentials* asset library. These can be customized into all kinds of custom brushes by duplicating them (see Brush Editing).

While it’s possible to have brush data-blocks that are local to the file and not marked as assets, such brushes cannot be activated for actual painting or sculpting. Use the *Mark as Asset* operator to make them brush assets that can be activated.

Brush Tool

Painting or sculpting with brushes requires the brush tool to be active. Activating a brush from an asset shelf or brush selector also activates the brush tool for convenience.



The *Brush* tool.

Manage Brushes

Brush assets are stored in asset libraries to make them accessible from any Blender session. There are two ways of managing brush assets:

- Using asset operators: Create and update brush assets using utility operators from any Blender file. Storage is managed by Blender. Convenient for simple “on the fly” management of personal brush asset libraries.
- Using manual storage: Create and update brush assets by opening blend files within asset libraries, and managing brush asset data-blocks in there. Useful for careful curation of asset libraries, especially to prepare them for sharing with others.

Asset Operators

Brushes can be managed through a few operators that let Blender handle the act of saving and updating the brushes in asset libraries for you. Assets managed this way will be saved in special asset system files using a *.asset.blend* file extension.

Note

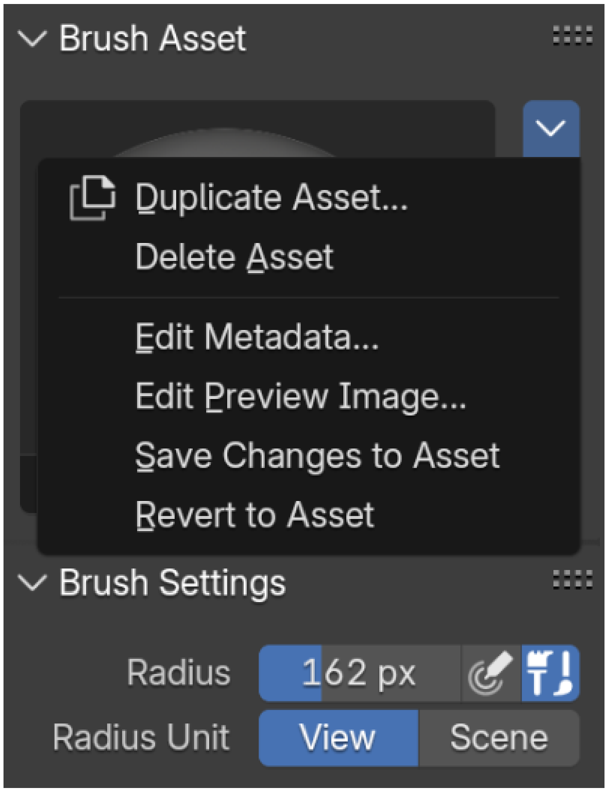
Note that only brush assets created via Duplicate Asset can be edited further using these asset operators. For others, these operations will be grayed out, and manual management is necessary.

Brushes from the *Essentials* asset library cannot be edited.

Reference

Mode::	All Paint Modes
Panel::	*Sidebar* ▸ *Tool* ▸ *Brush Asset Properties* ▸ *Tool* ▸ *Brush Asset*

Menu:: *Asset Shelf ‣ Context Menu*


Brush Asset
Duplicate Asset...
Delete Asset
Edit Metadata...
Edit Preview Image...
Save Changes to Asset
Revert to Asset
Brush Settings
Scene

Radius 162 px

Radius Unit View

Brush Asset panel in the Sidebar showing asset operators.

Duplicate Asset… (Duplicate icon)

Creates a copy of the currently active brush as asset, and activates it. A popup is spawned to input some settings to use:

Name

A custom name to use for the new brush.

Library

Choose an Asset Library to store the new brush asset in. The available asset libraries are configured in the Preferences.

Catalog

Choose an Asset Catalog to assign the brush asset to. Entering a non-existent name/path will create a new catalog accordingly.

Delete Asset

Permanently remove this brush asset from the Asset Library it is stored in. This cannot be undone, so a popup will ask for confirmation.

Edit Metadata…

Spawns a popup to change some of the available asset metadata fields:

Catalog

Choose an Asset Catalog to assign the brush asset to. Entering a non-existent name/path will create a new catalog accordingly.

Author

See Asset Author.

Description

See Asset Description.

Edit Preview Image…

Opens a window with the File Browser to select an image for the asset preview.

Save Changes to Asset

Saves any changes made to the active brush to the asset library.

Revert to Asset

Discards any unsaved changes made to the brush asset.

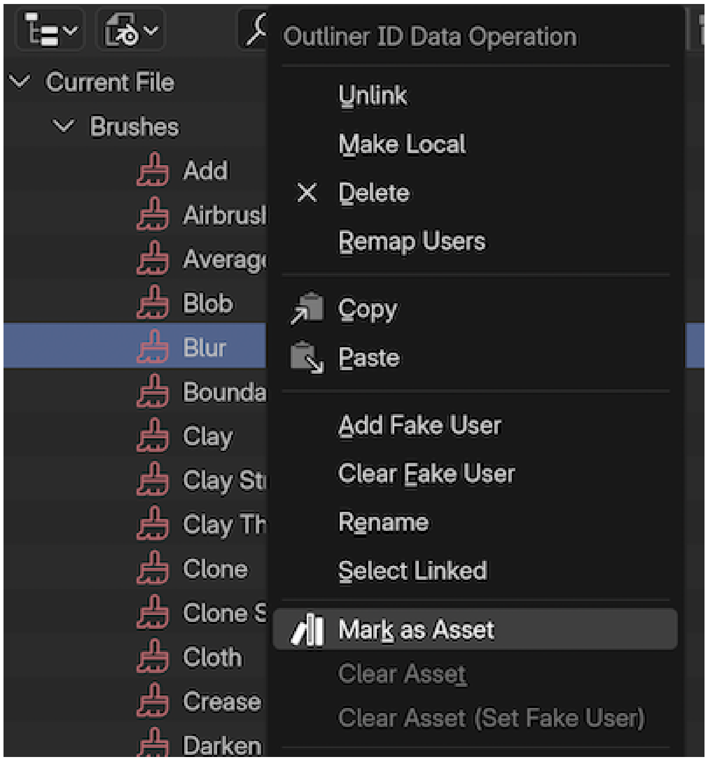
Manual Storage

See also

Life Cycle of an Asset

Complete description of the manual asset create, edit, share and use workflow.

It is also possible to manually manage brushes in blend-files like any other asset data-block. By marking brushes as assets and saving the file in an asset library, they become available from any Blender session. This gives full control over how assets are stored, and is particularly useful for curating asset libraries that can be shared with others.


Current File
Brushes
Add
Airbrusl
Averag
Blob
Blur
Bounda
Clay
Clay St
Clay Th
Clone
Clone S
Cloth
Crease
Darken
Outliner ID Data Operation
Unlink
Make Local
Delete
Remap Users
Copy
Paste
Add Fake User
Clear Fake User
Rename
Select Linked
Mark as Asset
Clear Asset
Clear Asset (Set Fake User)

The *Mark as Asset* operator used on a brush in the Outliner.

Brushes can be imported as normal data-blocks from other files (including from *.asset.blend* files from an asset library) through appending. In the Blender File mode of the Outliner, the brush will be listed under *Brushes*. Right-click the brush and select *Mark as Asset*. By saving the file inside of an asset library directory, the asset becomes available from all Blender sessions. If necessary, configure an asset library directory in the Preferences.

Brush Settings

Each mode and brush has unique brush settings. But there is also a lot of overlap or similar settings. This page explains general and mode specific settings that are used across various brushes in more detail.

General

Radius

> This option controls the size of the brush, measured in pixels. `F` allows you to change the brush size interactively by dragging the mouse from left to right and then `LMB` to accept. Meanwhile the texture of the brush will be visible inside the circle. You can also enter the size numerically with the number keys.
>
> The size can be decreased/increased using `[` and `]` respectfully.
>
> (Size Pressure)
>
> > Adjusts the radius based on the stylus pressure when using a Graphics Tablet.
>
> (Use Unified Radius)
>
> > Use the same brush *Radius* across all brushes.

Radius Unit *Sculpt Mode*

> Controls how the brush *Radius* is measured.

View::	The *Radius* is measured based on how the cursor appears on the monitor i.e. “screen space”.
Scene::	The *Radius* is measured based on real world units. This means that the brush radius stays consistent, independently from zooming in and out in the viewport. The unit type and scaling can be configured in the Scene Units.

Strength

For painting brushes the *Strength* defines the maximum effect of each brush stroke. For example, higher values cause a *Paint* brush to give each stroke a higher opacity. The opacity is never stronger than the set *Strength*, no matter how often the same surface is painted during the same stroke.

For sculpting brushes on the other hand the *Strength* relates to how strong each step of the stroke is, resulting in a slower/faster buildup towards the full brush effect during the stroke.

You can change the brush strength interactively by pressing `Shift-F` and then moving the brush and then `LMB`. You can also enter the strength numerically with the number keys.

(Strength Pressure)

Adjusts the strength based on the stylus pressure when using a Graphics Tablet.

(Use Unified Strength)

Use the same brush *Strength* across all brushes.

Blend

Set the way the color or value is applied over the targeted Color Attribute, Vertex Group or Image Texture. See Color Blend Modes.

- Add Alpha: makes the image more opaque where painted.
- Erase Alpha: makes the image transparent where painted, allowing background colors and lower-level textures to show through. As you “paint”, the false checkerboard background will be revealed. Using a tablet pen’s eraser end will toggle on this mode.

Tip

In order to see the effects of the Erase and Add Alpha mix modes in the Image Editor, the Display Channels must be set to *Color & Alpha* or *Alpha*. Transparent (no alpha) areas will then show a checkered background.

Weight *Weight Paint*

The weight value that is applied to the vertex group.

Use `Shift-X` to sample the weight value of clicked vertex. `Shift-Ctrl-X` lets you select the group from which to sample from.

Direction `Ctrl` *Sculpt Mode*

Brush direction toggle, *Add* raises geometry towards the brush, *Subtract* lowers geometry away from the brush. This setting can be toggled with `Ctrl` while sculpting.

Normal Radius *Sculpt Mode*

Determines the ratio of how much the brush radius is used to sample the normal direction of the sculpt plane of the brush. For example, a smaller *Normal Radius* will lead to drastic changes in the brush orientation, like for following the contours of hard surface meshes more closely. A large *Normal Radius* will lead to smoother changes in orientation, like for building overall forms on organic sculptures.

Area Radius

The ratio between the brush radius and the radius that is going to be used to sample the area plane depth.

Hardness *Sculpt Mode*

How close the brush falloff starts from the edge of the brush.

Tip Roundness

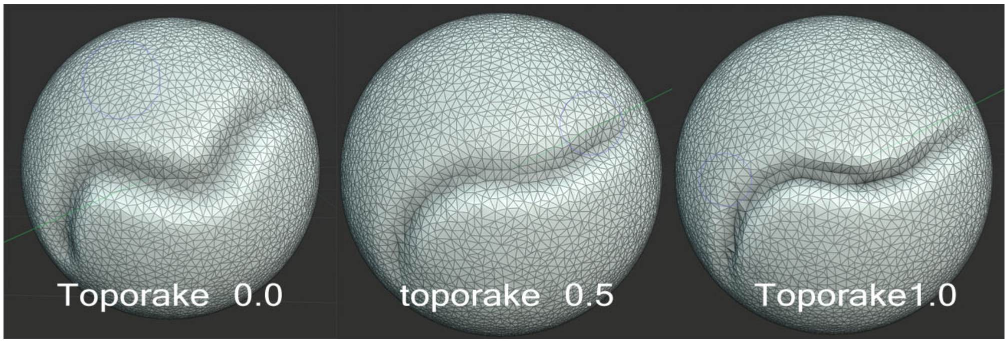
The factor to control how round the brush is. A value of zero will make the brush square. Note, the Brush Falloff is only applied to the rounded portions of the brush.

Auto-smooth *Sculpt Mode*

Sets the amount of smoothing to be applied to each stroke.

Topology Rake *Sculpt Mode*

The higher this setting is set, the more Dyntopo aligns mesh edges to the brush direction while tessellating the surface. This generates cleaner edge flow to help define sharp features. *Topology Rake* can have a severe performance impact so it works best on low-poly meshes.



Normal Weight `Ctrl` *Sculpt Mode*

Constrains brush movement along the surface normal. Especially useful with the *Grab* brush, can be temporarily enabled by holding `Ctrl`. E.g. *Grab* brush can be used to push a depression (hole) into the mesh when *Normal Weight* is set.

Applies to *Grab* and *Snake Hook* brushes.

Plane Offset *Sculpt Mode*

Offset for planar brushes (Clay, Fill, Flatten, Scrape), shifts the plane that is found by averaging the faces above or below.

Plane Trim *Sculpt Mode*

Ability to limit the distance that planar brushes act. If trim is enabled vertices that are further away from the offset plane than the trim distance are ignored during sculpting.

Pinch/Magnify *Sculpt Mode*

Pushes the mesh towards/away from the brush center during the stroke.

Deformation Target

How the deformation of the brush will affect the object.

Geometry::	Deform the geometry directly.
Cloth Simulation::	Deform the mesh while a cloth simulation is applied to it at the same time.

Advanced

Brush Type

Defines the basic behavior and the available settings. Through the settings of a brush type, brushes can be created that produce vastly different effects.

The *Essentials* asset library contains brushes for each of the brush types. Their preview image and description should give a good idea of the effect the brush produces, with the particular combination of brush type and settings. Because of this, they are usually the more useful starting point for custom brushes than the mere brush type is, which is why the brush type is part of the *Advanced* brush settings.

Brushes and Brush Types of each mode:

- Sculpt
- Vertex Paint
- Weight Paint
- Texture Paint

Accumulate

Causes stroke dabs to accumulate on top of each other.

Front Faces Only

When enabled, the brush only affects vertices that are facing the viewer.

Affect Alpha *2D Painting Only*

When this is disabled, it prevents changes to the alpha channel while painting (Only in 3D Viewport).

Anti-Aliasing *2D Painting Only*

Toggles Anti-Aliasing around the brush, this is useful if you are working with pixel art or low resolution textures.

Auto-Masking *Sculpt Mode*

The auto-masking toggles in the brush settings are the same as the sculpt mode auto-masking settings. The difference is that these toggles can be customized per brush to create specific brush behaviors.

See also

For more information on the Auto-Masking toggles, see Auto-Masking.

Sculpt Plane *Sculpt Mode*

Use this menu to set the plane in which the sculpting takes place. In other words, the primary direction that the vertices will move.

Area Plane::	The movement takes place in the direction of average normal for all active vertices within the brush area. Essentially, this means that the direction is dependent on the surface beneath the brush.
View Plane::	Sculpting in the plane of the current 3D Viewport.
X, Y, Z Plane::	The movement takes place in the positive direction of one of the global axes.

Original *Sculpt Mode*

Normal

When locked it keeps using the normal of the surface where stroke was initiated, instead of the surface normal currently under the cursor.

Plane

When locked keep using the plane origin of surface where stroke was initiated, instead of the surface plane currently under the cursor.

Color Picker

Color

The color of the brush. See Color Picker.

Press `Shift-X` on any part of the image to sample that color and set it as the brush color. Hold `Ctrl` while painting to temporally paint with the secondary color.

Swap Colors (cycle icon) `X`
  Swaps the primary and secondary colors.

(Use Unified Color)
  Use the same brush color across all brushes.

Note

Note that Vertex Paint works in sRGB space, and the RGB representation of the same colors will be different between the paint tools and the materials that are in linear space.

Gradient

A gradient can be used as a color source.

Gradient Colors
  The Color Ramp Widget to define the gradient colors.

Mode

Pressure::	Will choose a color from the color ramp according to the stylus pressure.
Clamp::	Will alter the color along the stroke and as specified by *Gradient Spacing* option. With *Clamp* it uses the last color of the color ramp after the specified gradient.

Repeat::	Similar to *Clamp*. After the last color it resets the color to the first color in the color ramp and repeats the pattern.

Color Palette

Color Palettes are a way of storing a brush's color so that it can be used at a later time. This is useful when working with several colors at once.

Palette
A Data-Block Menu to select a palette.

New +
Adds the current brush's primary *Color* to the palette.

Delete –
Removes the currently selected color from the palette.

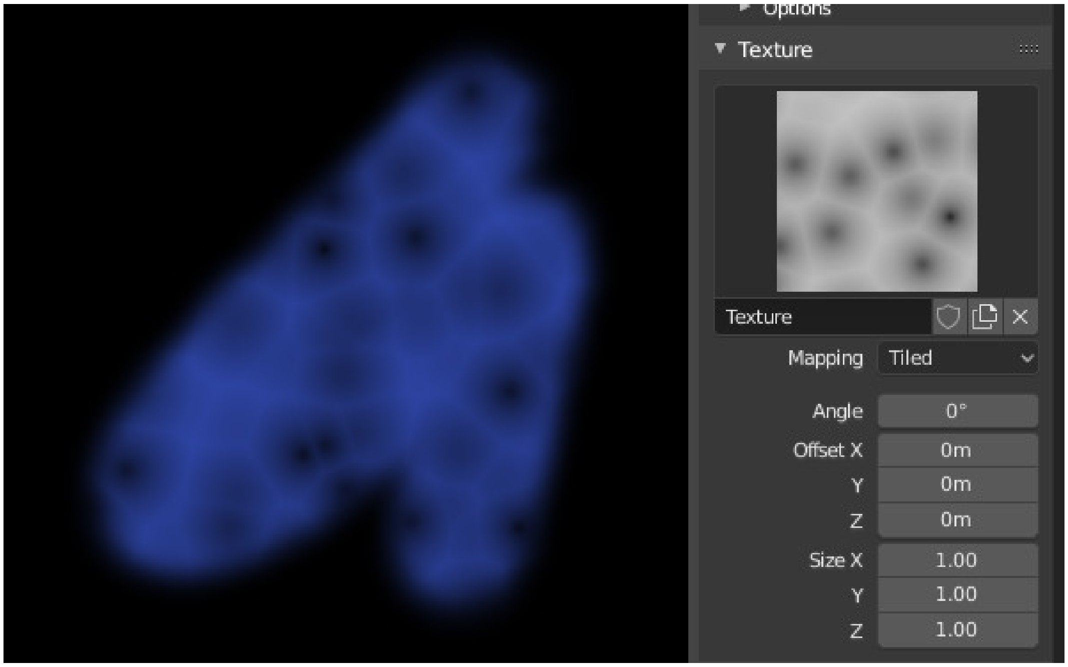
Move (up/down arrow icon)
Moves the selected color up/down one position.

Sort
Sort Colors by Hue, Saturation, Value, Luminance.

Color List
Each color that belongs to the palette is presented in a list. Clicking on a color will change the brush's primary *Color* to that color.

Texture & Texture Mask

This page covers both the Texture and Texture Mask panels. Add a Texture to the brush to control the color of the brush. A Texture Mask is used to control the strength of the brush. Both the Texture and Texture Mask offer the same settings.



Texture

Mapping Tiled

Angle 0°

Offset X 0m

Y 0m

Z 0m

Size X 1.00

Y 1.00

Z 1.00

Example of the Texture panel and a textured brush in use.

Texture

In paint modes the texture is used as a color source, while for sculpting it is used to determine the strength of the brush.

Any image texture or procedural texture can be assigned in the texture and texture mask panels. Textures can be further edited in the properties editor (Click the properties icon for quick access)

Tip

It’s recommended to load all needed images ahead of time as image textures into Blender. Then they can be easily selected by clicking on the texture and picking it from the data-block popup. Textures can also be appended/linked from other Blender files.

Mapping & Mask Mapping

How the texture is applied to the brush stroke.

Tip

It is recommended to set this to *Area Plane* or *View Plane* for the most common behavior. Ideally match this setting with the Sculpt Plane setting if in sculpt mode.

View Plane::	If *View Plane* is enabled, the current viewing angle is used to project the brush texture onto the model. This is especially useful for projection painting.
Area Plane::	Projects the brush texture along the local surface normal, which keeps the texture from stretching on steep angles. This is an ideal default for most brushes.
Tiled::	The *Tile* option repeats the texture across the screen, so moving the brush will not change where the texture is applied. The *Tile* option is most useful with tileable images, rather than procedural textures.
3D::	The *3D* option allows the brush to take full advantage of procedural textures. This mode uses vertex coordinates rather than the brush location to determine what area of the texture to use. This option is not available for the Texture Mask.
Random::	Picks a random texture coordinate to sample from for each step of the stroke.

Stencil:: This is the ideal option for stamping textures for projection painting. Stencil mapping works by projecting the texture from the camera space on the mesh or canvas. Painting is applied only inside the boundaries of the stencil. The stencil is displayed as a screen space overlay on the viewport. To transform the stencil texture use the following shortcuts (Hold `Alt` for the Texture Mask):

- Move `RMB`
- Scale `Shift-RMB`
- Rotate `Ctrl-RMB`

While using stencil scaling, `X` and `Y` are used to constrain the scaling to one axis. Pressing one of the buttons twice reverts to unconstrained scaling.

Image Aspect

Restore the aspect ratio of the original image to reset stretching introduce by scaling, (Image textures only.) This operator can use the tiling and scale values of the brush texture if the relevant are enabled in Adjust Last Operation panel.

Reset Transform

Restores the position of the stencil.

Pressure Masking

Only available for the Texture Mask. It allows to clip the mask result based on pressure.

Off::	Disabled.
Ramp::	Fades out the mask effect on higher pressure.
Cutoff::	Expands the used values from the image based on stylus pressure.

Angle `Ctrl-F`

This is the rotation angle of the texture brush. It can be changed interactively via `Ctrl-F` in the 3D Viewport. While rotating the angle via the hotkey you can enter a value numerically as well.

Rake

Texture angle follows the direction of the brush stroke. Useful for stamping textures repeatedly along the stroke. Not available with *3D*, *Tiled*, or *Stencil* Mapping types. The shortcut is not available in Sculpt mode.

Random

Angle is randomized on each step of the stroke. Not available with *3D*, *Tiled*, or *Stencil* Mapping types. The shortcut is not available in Sculpt mode.

Random Angle

Constraints the random deviation to a range.

Offset X, Y, Z

Offset the texture map placement in X, Y, and Z axes.

Size X, Y, Z

Set the scale of the texture in each axis.

Sample Bias *Sculpt Mode*

Value added to texture samples. This can be used if the mid-level of a height map is not correct.

Vector Displacement *Sculpt Mode*

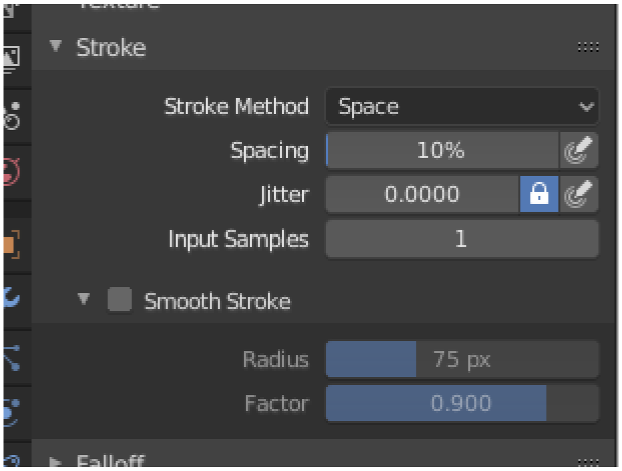
Use the color channels to displace geometry in 3 vectors.

Note

This is only supported for the Draw brush with Area Plane mapping enabled.

Stroke

The stroke settings define the behavior of the sculpted/painted stroke. Any other brush behavior and effect is applied on top of the stroke.



Stroke Method: Space
Spacing: 10%
Jitter: 0.0000
Input Samples: 1
Radius: 75 px
Factor: 0.900

Stroke panel.

Stroke Method `Alt-E`

Defines the way brush strokes are applied to the canvas.

Dots::	Apply paint on each mouse move step. This is regardless of their distance to each other, and instead depends on the stroke speed. This means that a slower stroke will have more accumulative strength applied.
Drag Dot::	Leaves only one dab on the canvas which can be placed by dragging.
Space::	Creates brush stroke as a series of dots, whose distance (spacing) is determined by the *Spacing* setting. Spacing Limits brush application to the distance specified by the percentage of the brush

radius.

(Spacing Pressure)

Brush *Spacing* can be affected by enabling the pressure sensitivity icon, if you are using a Graphics Tablet.

Airbrush:: Flow of the brush continues as long as the mouse click is held (spray), determined by the *Rate* setting.

Rate

Interval for how frequent the brush is applied during the stroke.

Anchored:: Creates a single dab at the brush location. Clicking and dragging will resize the dab diameter.

Edge to Edge

The brush location and orientation are determined by a two point circle, where the first click is one point, and dragging places the second point, opposite from the first.

Line:: Clicking and dragging lets you define a line in screen space. The line dabs are separated by *Spacing*, similar to space strokes. With `Alt` the line stroke is constrained to 45 degree increments.

Curve:: Defines the stroke curve with a Bézier curve (dabs are separated according to *Spacing*). This Bézier curve is stored in Blender as a “Paint Curve” data-block.

Use `Ctrl-RMB` to create the initial control point of the curve.

Paint Curves

Paint Curves are reusable and can be stored and selected by using the Data-Block Menu menu.

- **Add Points**

  You can define additional curve control points by using `Ctrl-RMB`. The handles can be defined by dragging the mouse. The stroke flows in the direction of the first control point to the second control point, and so on.

- **Transforming Points**

  The control points and handles can be dragged with `RMB` (In right click select with `LMB`). To make sure that the handles of a control point are symmetrical, drag them using `Shift-RMB`. A few transform operators are supported such as moving (`G`), rotating (`R`) and scaling (`S`).

- **Selection**

  The handles can be selected individually by using `LMB` (In right click select with `RMB`), extend the selection by `Shift-LMB` and deselect/select all by using `A`.

- **Delete Points :kbd:`X`**

  To delete a curve point, use `X`.

Draw Curve `Return`

To confirm and execute the curved stroke, press `Return` or use the Draw Curve button.

Spacing Distance *Sculpt Mode Only*

Method used to calculate the distance to generate a new brush step.

View::	Calculates the brush spacing relative to the view.
Scene::	Calculates the brush spacing relative to all three dimensions of the scene using the stroke location. This avoids artifacts when sculpting across curved surfaces and keeps the spacing much more consistent.

Adjust Strength for Spacing

Keep the brush strength consistent, even if the spacing changes. Available for the *Space*, *Line*, and *Curve* stroke methods.

Dash Ratio

Ratio of samples in a cycle that the brush is enabled. This is useful to create dashed lines in texture paint or stitches in Sculpt Mode. Available for the *Space*, *Line*, and *Curve* stroke methods.

Dash Length

Length of a dash cycle measured in stroke samples. This is useful to create dashed lines in texture paint or stitches in Sculpt Mode. Available for the *Space*, *Line*, and *Curve* stroke methods.

Jitter

Jitter the position of each step in the brush stroke.

(Jitter Pressure)

Brush *Jitter* can be affected by enabling the pressure sensitivity icon, if you are using a Graphics Tablet.

Jitter Unit

Controls how the brush *Jitter* is measured.

View::	The *Jitter* is relative to the view direction i.e. “screen space”.
Scene::	The *Jitter* is measured relative to all three dimensions of the scene. The unit type and scaling can be configured in the Scene Units.

Input Samples

Recent mouse locations (input samples) are averaged together to smooth brush strokes.

Use Unified Input Samples

Use the same brush *Input Samples* across all brushes.

Stabilize Stroke

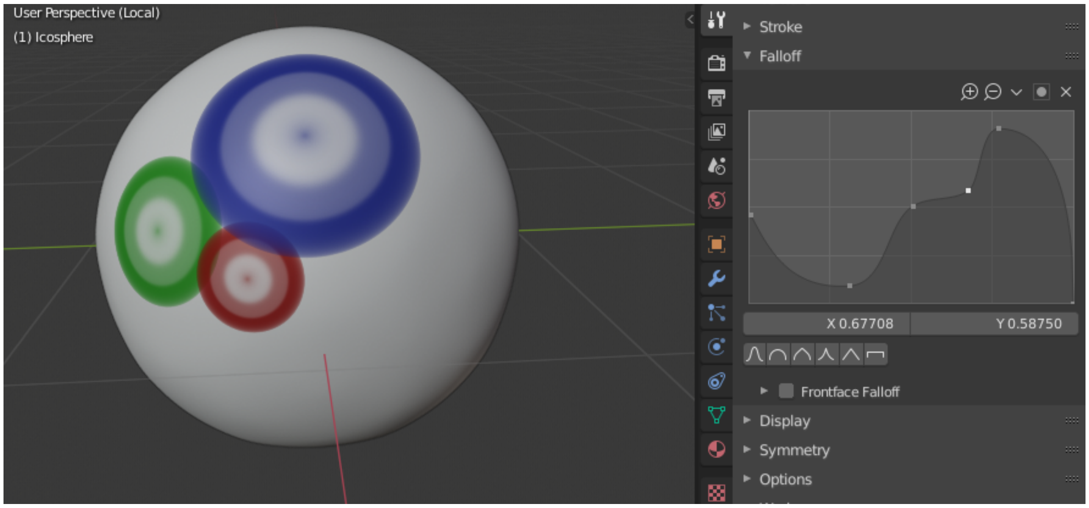
Stabilize Stroke makes the stroke lag behind the cursor and creates a smoothed curve to the path of the cursor. This can be enabled pressing `Shift S` or by clicking the checkbox found in the header.

Radius
: Minimum distance from the last point before the stroke continues.

Factor
: A smooth factor, where higher values result in smoother strokes but the drawing sensation feels like as if you were pulling the stroke.

Falloff

The Falloff allows you to control the *Strength* falloff of the brush. The falloff is mapped from the center of the brush (left part of the curve) towards its borders (right part of the curve). Changing the shape of the curve will make the brush softer or harder. Read more about using the Curve Widget.

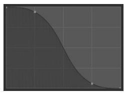

User Perspective (Local)
(1) Icosphere
Stroke
Falloff
X 0.67708
Y 0.58750
Frontface Falloff
Display
Symmetry
Options

Brush curve example.

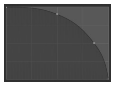
Curve Preset

Custom:: You can choose how the strength of the falloff is determined from the center of the brush to the borders by manually manipulating the control points within the curve widget. There are also a couple of preset custom curves displayed at the bottom of the curve widget that can be used on their own or as a starting point for tweaking.

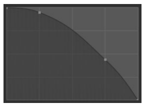
Custom Preset types.



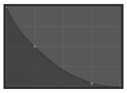
Smooth.

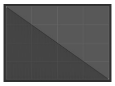


Sphere.

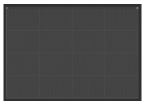


Root.





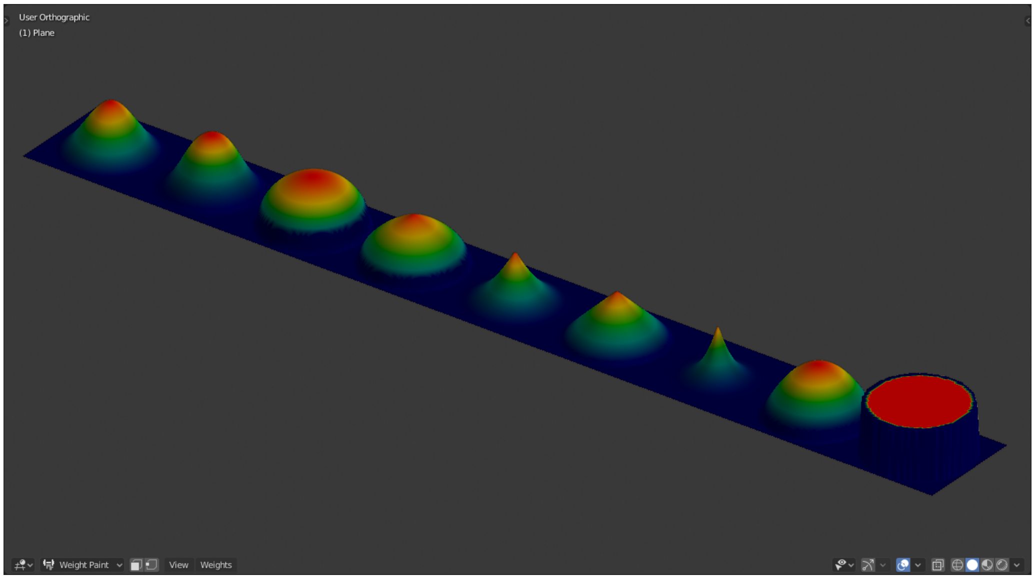
Linear.



Sharp.

Constant.

Smooth::	The center strength, the border strength, and the falloff transition between them are evenly distributed.
Smoother::	Similar to *Smooth* but produces a wider center point of the brush before tapering off.
Sphere::	The strength of the brush is predominately at its strongest point with a steep falloff near the border of the brush.
Root::	Similar to a Sphere but the center is a more concentrated point.
Sharp::	The center of the brush is the strongest point then exponentially tapers off to a lower strength, creating a fine point.
Linear::	With the center being the strongest, the strength will consistently weaken as it reaches the border of the brush.
Sharper::	Similar to *Sharp* but the center point is more condensed.
Inverse Square::	A hybrid between *Smooth* and *Sphere*.
Constant::	The strength of the brush remains unified across the entire brush. This will create a sharp edge at the border of the brush.



(From Left to Right) Smooth, Smoother, Sphere, Root, Sharp, Linear, Sharper, Inverse square, Constant.

Falloff Shape

Use projected or spherical falloff. Note, this is not supported in Texture Paint Mode.

Sphere::	Applies brushes influence in a sphere, outwards from the center.
Projected::	This turns the brush influence into a cylinder (the depth along the view is ignored) instead of a sphere. It can be used along the outline of a mesh to adjust its silhouette.

Front-Face Falloff

As faces point away from the view the brush strokes fade away to prevent harsh edges.

Normal Falloff / Front-Face Falloff

If disabled, the normal of the surface has no effect on the falloff.

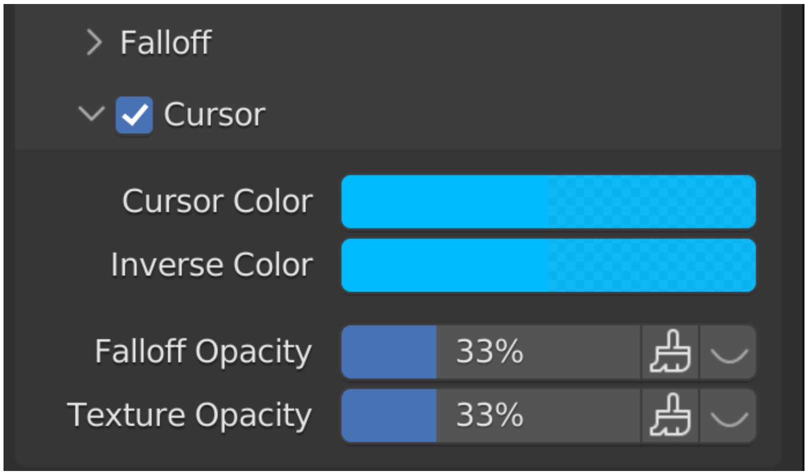
Angle

The angle at which the falloff begins.

Cursor

Reference

Mode::	All Paint Modes
Header::	*Tool Settings ‣ Brush Settings ‣ Cursor*
Panel::	*Sidebar ‣ Tool ‣ Brush Settings ‣ Cursor*



Cursor options.

While painting or sculpting a special cursor is shown to display information about the active brush. The cursor is shown as a circle in the 3D Viewport, the radius of the circle matches the size of the brush.

The cursor can be disabled by toggling the checkbox in the panel's header.

Cursor Color
: Set the color of the brush ring while performing an add/positive stroke.

Inverse Color
: In some paint/sculpt modes the brush can be negative and subtract information from the paint target; these brushes can be given a separate color.

Opacity Options
: Depending on the paint or sculpt mode different overlays are shown within the cursor to give information on how the brush is textured. This

is most commonly used to show the brush falloff with a gradient from the circle center to the perimeter.

Alpha

You can change the amount of transparency used when showing the texture using the slider.

Override Overlay (brush icon)

Allows you to turn off the viewport overlay during strokes.

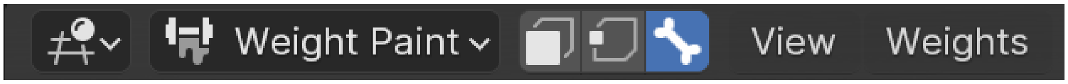
View (eye icon)

Toggles whether to show or hide the given brush texture overlay.

Selection & Visibility

Selection Masking

If you have a complex mesh, it is sometimes not easy to paint on the intended vertices. Suppose you only want to paint on a small area of the Mesh and keep the rest untouched. This is where “selection masking” comes into play. When this mode is enabled, a brush will only paint on the selected vertices or faces. The option is available from the header of the 3D Viewport (see icons surrounded by the yellow frame):



> You can choose between *Face Selection masking* (left button), *Vertex selection masking* (middle button), and *Bone selection* (right button). The latter is only available when the mesh has an Armature modifier.

Selection masking has some advantages over the default paint mode:

- The original mesh edges are shown, even when modifiers are active.
- You can select and deselect faces instead without the need to switch to Edit Mode.

Details About Selecting

The following standard selection operations are supported:

- `Alt-LMB` – Single faces
- `Shift-Alt-LMB` – Select more or remove them from the selection.
- `A` – All faces, `A A` to deselect.
- `B` – Box selection.
- `C` – Circle select with brush.
- `Ctrl-I` – Invert selection.

- `L` – Pick linked (under the mouse cursor).
- `Ctrl-L` – Select linked.
- `Ctrl-NumpadPlus` – Extend Selection
- `Ctrl-NumpadMinus` – Shrink Selection

The following only work for face selection and with the selection tool active:

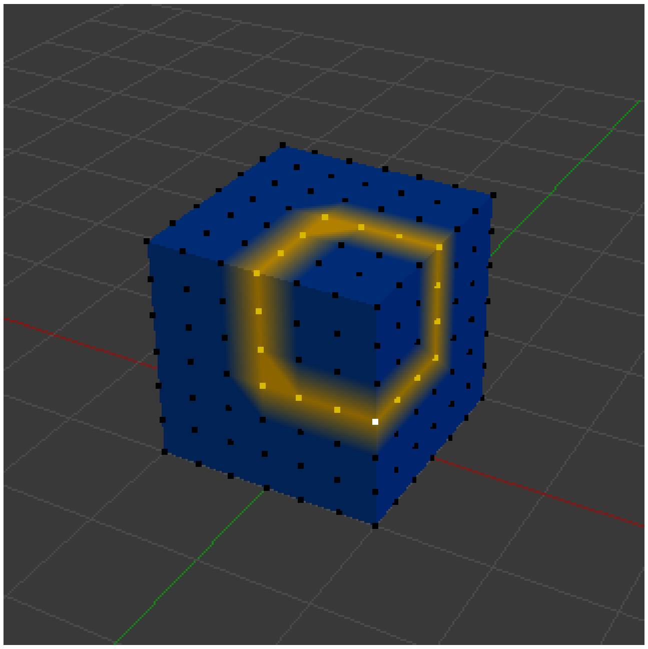
- `Alt-LMB` – Loop Select

Vertex Selection Masking

Reference

Mode::	Vertex and Weight Paint Modes
Header::	*Vertex Selection*
Shortcut::	`2`

In this mode you can select one or more vertices and then paint only on the selection. All unselected vertices are protected from unintentional changes.



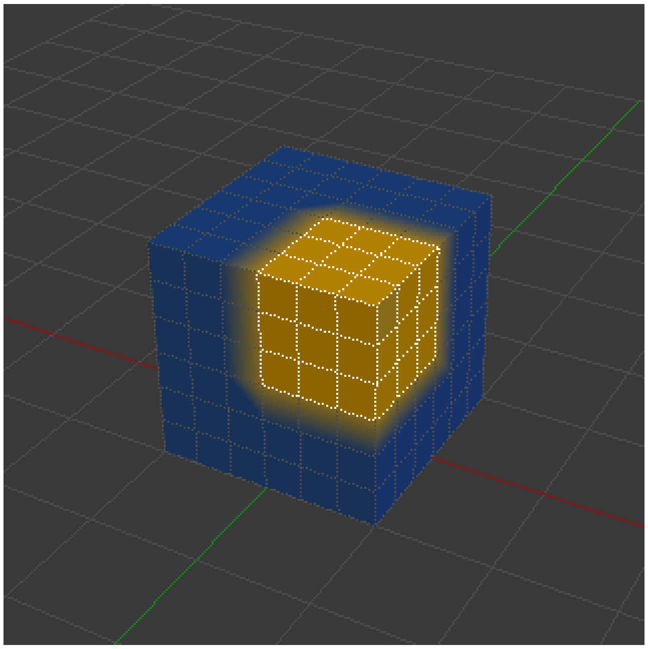
Vertex Selection masking.

Face Selection Masking

Reference

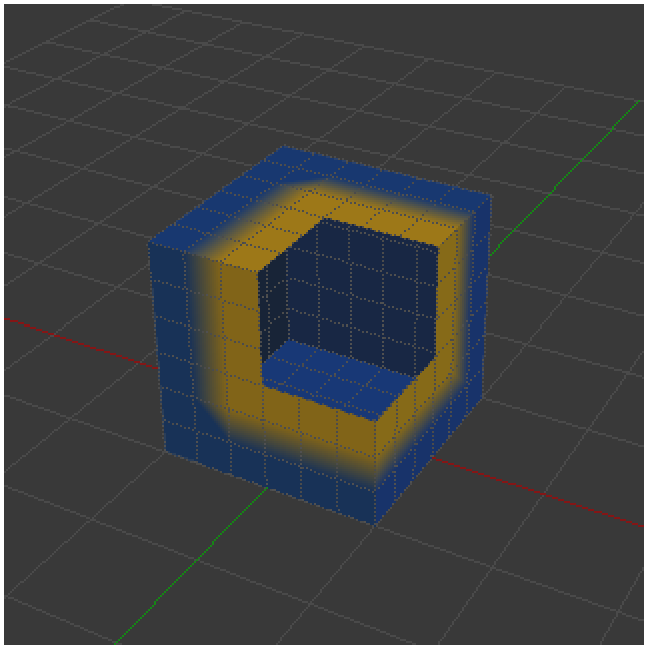
Mode:: Texture, Vertex, and Weight Paint Modes
Header:: *Paint Mask*
Shortcut:: `1`

The *Face Selection masking* allows you to select faces and limit the paint tool to those faces, very similar to Vertex selection masking.



Face Selection masking.

Hide/Unhide Faces



Hidden faces.

You also can hide selected faces as in Edit Mode with the keyboard Shortcut `H`, then paint on the remaining visible faces and finally unhide the hidden faces again by using `Alt-H`.

Hide/Unhide Vertices

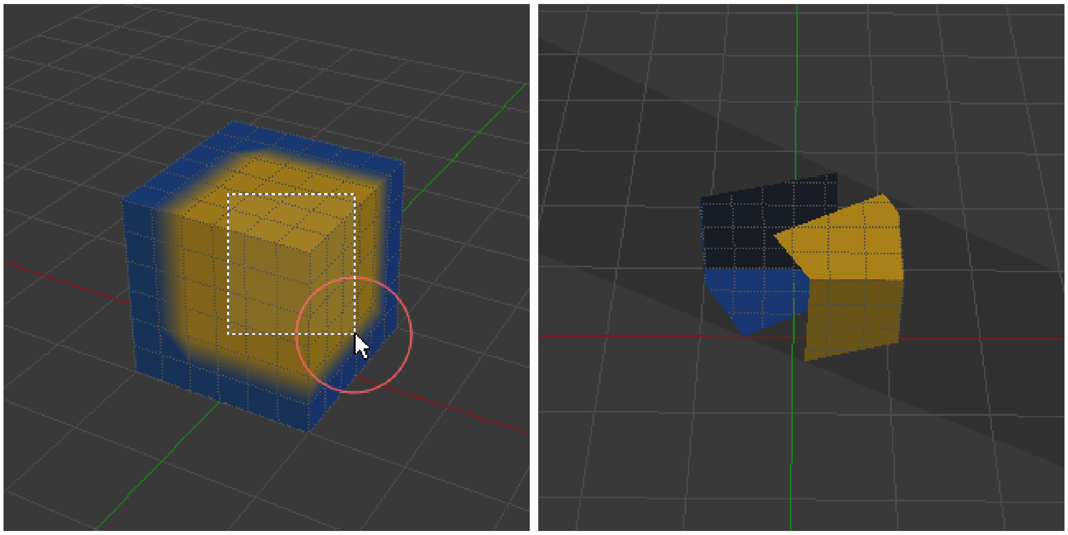
You cannot specifically hide only selected faces in vertex mask selection mode. However, the selection is converted when switching selection modes. So a common trick is to:

1. Switch to Face selection mask mode to have the selection converted to faces.
2. Refine your selection next or just hide the faces.
3. Switch back to Vertex Selection mask mode.

Hidig faces will make sure that vertices that belong to visible faces remain visible.

The Clipping Region

To constrain the paint area further you can use the *Clipping Region*. Press `Alt-B` and `LMB`-drag a rectangular area. The selected area will be "cut out" as the area of interest. The rest of the 3D Viewport gets hidden.



The Clipping Region is used to select interesting parts for local painting.

You make the entire mesh visible again by pressing `Alt-B` a second time.

All paint tools that use the view respect this clipping, including box select, and of course brush strokes.

There are two helpful reminders that a Clipping Region is used:

1. The clipping region is drawn as a grey box in the 3D Viewport
2. The Text Info overlay will state that the perspective is "Clipped"

Select Linked

Reference

Mode::	Edit Mode
Menu::	*Select ‣ Select Linked ‣ Linked*
Shortcut::	`Ctrl-L`, `Shift-L`

Select geometry connected to already selected elements. This is often useful when a mesh has disconnected, overlapping parts, where isolating it any other way would be tedious. Pressing `Shift–L` will deselect linked any linked elements.

With `L` you can also select connected geometry directly under the cursor.

Navigation

There are different preferences for navigating the 3D Viewport in Blender. For painting and sculpting specific workflows it is recommended to use any of the following methods.

Center View to Mouse `Alt MMB`

Center the View on the surface directly under the mouse position. This way the rotation point of the viewport can be manually changed to any point you wish to orbit around.

Center on Last Stroke `NumpadPeriod`

Center the View on the average position of the last stroke.

Various preferences can also make navigation more convenient. These can be found in the "Navigation" tab of the preferences.

Orbit Method = Trackball

Tumble the view based on the mouse position in your 3D Viewport while rotating. This makes it very easy to tilt the viewport freely, instead of having the Z axis of the viewport locked.

Zoom to Mouse Position

Use the mouse position to zoom towards and rotate around the surface that is pointed at. This can be an alternative to the repeated manual use of the Center View to Mouse operator.

The disadvantage is that this navigation preference can lead to accidental navigation around backfacing geometry or very distant geometry.

Sculpting

- Introduction
    - General
    - The Brush
    - Gesture Tools
    - Visibility, Masking & Face Sets
    - Filters
    - Transforming
    - Painting
    - Working with Multiple Objects
    - Adaptive Resolution
    - Cloth Sculpting
- Brushes
    - Brushes
    - Draw
    - Draw Sharp
    - Clay
    - Clay Strips
    - Clay Thumb
    - Layer
    - Inflate
    - Blob
    - Crease
    - Smooth
    - Flatten
    - Fill
    - Scrape
    - Scrape Multiplane
    - Pinch
    - Grab
    - Elastic Deform
    - Snake Hook
    - Thumb
    - Pose

  - Nudge
  - Rotate
  - Relax Slide
  - Boundary
  - Cloth
  - Density
  - Mask
  - Draw Face Sets
  - Erase Multires Displacement
  - Smear Multires Displacement
  - Paint
  - Smear
- Toolbar
  - Sculpting Tools
  - Gesture Tools
  - Filter Tools
  - Single Click Tools
  - General Tools
- Tools
  - Brush
  - Mask Gesture Tools
  - Hide Gesture Tools
  - Face Set Gesture Tools
  - Trim Gesture Tools
  - Line Project
  - Mesh Filter
  - Cloth Filter
  - Color Filter
  - Edit Face Set
  - Mask by Color
  - Transforms
- Tool Settings
  - Brushes
  - Brush Settings
  - Dyntopo
  - Remesh
  - Symmetry

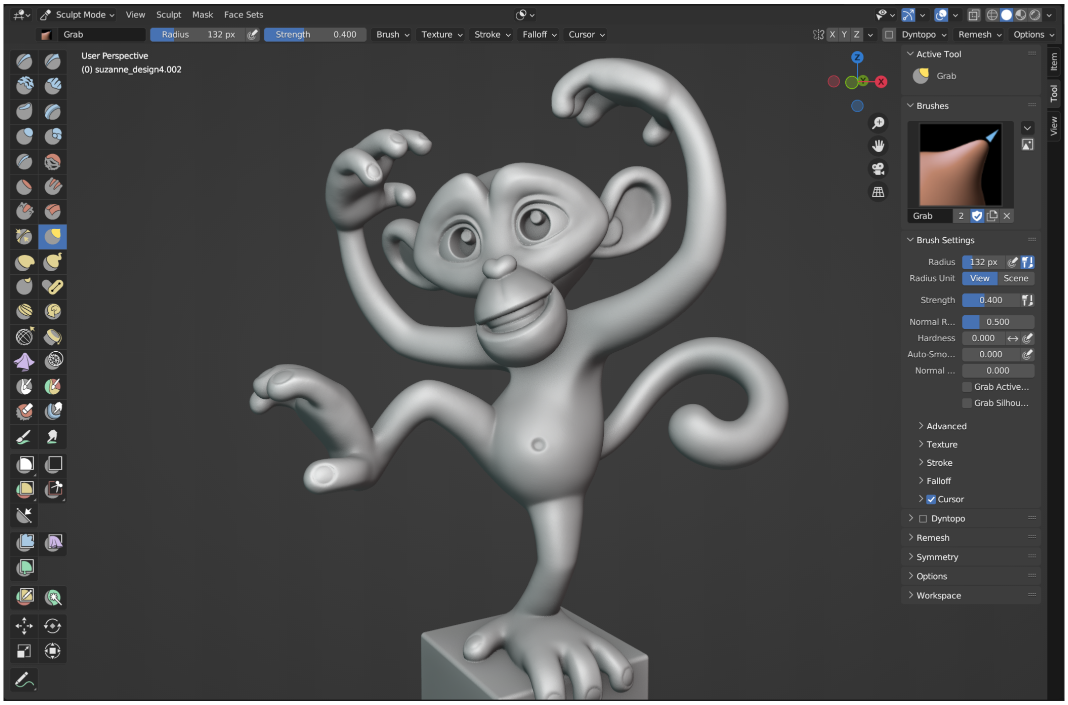
  - Options
- Controls
  - Auto-Masking
- Editing
  - Sculpt
  - Mask
  - Face Sets
  - Expand

Introduction

- General
- The Brush
  - Common Brushes
- Gesture Tools
  - Box Gestures
  - Lasso Gestures
  - Line Gestures
  - Polyline Gestures
- Visibility, Masking & Face Sets
  - Visibility Control
  - Masks
  - Face Sets
  - Auto-Masking
  - Display Settings
- Filters
- Transforming
- Painting
- Working with Multiple Objects
- Adaptive Resolution
  - Voxel Remesher
  - Dyntopo
  - Multiresolution
- Cloth Sculpting

General

Sculpt Mode is similar to *Edit Mode* in that it is used to alter the shape of a model, but Sculpt Mode uses a very different workflow: instead of dealing with individual elements (vertices, edges, and faces), an area of the model is primarily changed using brushes.


Sculpt Mode
View
Sculpt
Mask
Face Sets
Grab
Brush
Texture
Stroke
Falloff
Cursor
X Y Z
Dyntopo
Remesh
Options
User Perspective
(0) suzanne_design4.002
Active Tool
Grab
Brushes
Grab 2
Brush Settings
Grab Active...
Grab Silhou...
Advanced
Texture
Stroke
Falloff
Cursor
Dyntopo
Remesh
Symmetry
Options
Workspace
Item
Tool
View

Radius 132 px
Radius Unit View Scene
Strength 0.400
Normal R... 0.500
Hardness 0.000
Auto-Smo... 0.000
Normal ... 0.000

Sculpting Mode Example.

Sculpt Mode is accessed from the mode menu of the [3D Viewport header]() or with the pie menu via `Ctrl-Tab`. Once inside Sculpt Mode, the Toolbar and Tool Settings of the 3D Viewport will change to Sculpt Mode specific panels. The cursor will change to a circle, to indicate the size of the brush.

Warning

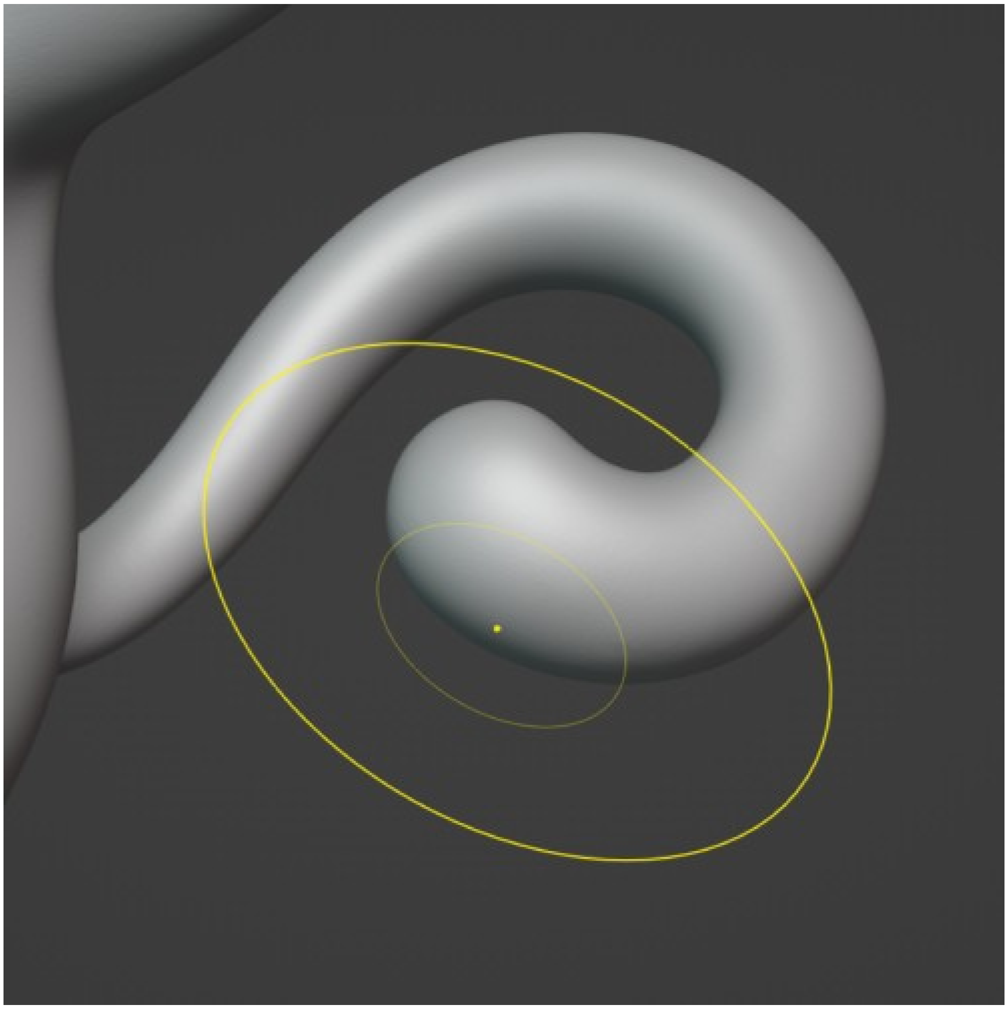
To have predictable brush behavior, make sure to apply the scale of your mesh.

The following pages will briefly explain the fundamental features and concepts of *Sculpt Mode*, including various links to other pages for more details.

The Brush

Sculpt Mode is very recognizable by the behavior and visualization of the brush. All the usual brush controls still apply, yet the brush for sculpting is displayed in 3D. This means that the brush will follow the curvature of the surface by orienting the radius to match the topology Normal.

The inner ring of the brush cursor is used to visualize the strength of the brush.



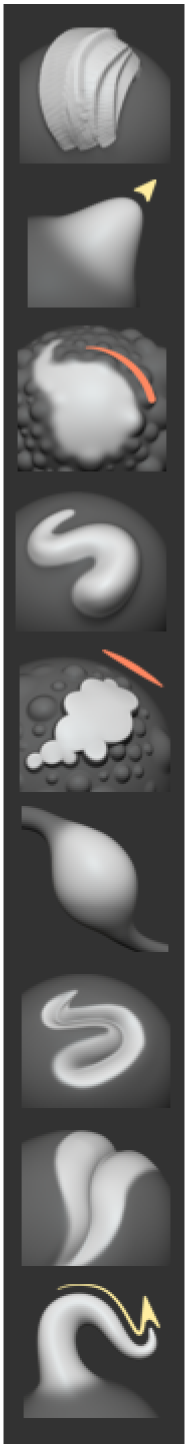
Note

How closely the cursor follows the curvature of the mesh can be changed in the Brush Settings with “Normal Radius”. This can make hard surface sculpting easier, for example with the Scrape brush.

The brush is also used for other tools in the toolbar to better display how that tool works. For example, the Box Trim and Lasso Trim tools are able to use the current brush radius for how deep geometry is trimmed or added.

Common Brushes

There are many brushes to choose from but these are the most common brushes to be used during sculpting. More information on sculpting brushes in the Toolbar.



Clay Strips

Block out broad shapes and build up volumes before refining them further.

Grab

Move geometry across the screen for general shaping.

Smooth
Smooth and shrink surfaces to remove noise or flatten shapes.

Draw
Generic adding and subtracting on surfaces. This brush is often customized with different stroke methods and textures for various effects.

Scrape
Scrape and fill surfaces either for hard surface sculpting or more aggressive smoothing.

Inflate
Inflate or shrink volumes or surfaces. Especially useful for controlling the thickness of cylindrical shapes.

Draw Sharp
Same as *Draw* but with a much sharper falloff. Useful for adding creases, cracks and other sharp edges.

Crease
A mix of the *Draw* and *Pinch* brushes. Useful for creating detailed creases or sharpening existing creases for additional polish.

Snake Hook.
Similar to *Grab* but this brush will dynamically let go and pick up geometry during the stroke. The dragged geometry is also following the angle of the stroke, making it very useful for pulling geometry out. Ideally used together with Dyntopo.

Gesture Tools

Separate from brushes and filters, Sculpt mode also has a set of tools that perform actions to a drawn selection area. These tools are similar to the selection tools (e.g. box selection and lasso selection in other areas of Blender).

These tools do not provide a selection of elements that are then modified, they directly modify the underlying mesh.

Box Gestures

Dragging creates a rectangular area defined by where `LMB` was pressed and where `LMB` is released.

Controls

Move `Spacebar`
Hold to reposition the selection area.

Lasso Gestures

Dragging creates a freeform area that follows the cursor defined by where `LMB` was pressed and where `LMB` is released.

Controls

Move `Spacebar`
Hold to reposition the selection area.

Tool Settings

Stabilize Stroke

Helps to reduce jitter of the strokes while drawing by delaying and correcting the location of points.

Radius
: Minimum distance from the last point before the stroke continues.

Factor
: A smooth factor, where higher values result in smoother strokes but the drawing sensation feels as if you were pulling the stroke.

Line Gestures

Dragging creates a line. The resulting action acts upon everything on the highlighed side of the line. The area acted upon is extended in both directions of the viewport.

Controls

Flip `F`
: Toggles the side of the line that the tool affects.

Snap `Ctrl`
: Hold to constrain the rotation of the line to user-specified intervals. Defaults to 5 degree increments, customizable via the Snapping menu indicated by the magnet icon in the header.

Move `Spacebar`
: Hold to reposition the line.

Tool Settings

Limit to Segment
: The affected area will not extend the length of the drawn line. This helps defining a smaller area instead of extending the line infinitely long.

Polyline Gestures

Clicking places a point in the viewport. Each time `LMB` is pressed, a new point of the polygon is created. Pressing `LMB` on the starting point, pressing `LMB` twice, or pressing `Return` closes the selection area.

Controls

Move `Spacebar`

Hold to reposition the selection area.

Visibility, Masking & Face Sets

Visibility Control

Parts of the mesh can be hidden in Sculpt Mode. Because hidden faces cannot be sculpted, hiding makes it easier to isolate what you want to work on. Hiding geometry also improves the viewport performance.

Hiding is shared between all modes, except Object Mode (i.e. hiding/showing of faces in one mode will hide the same faces in other modes too).

Unlike Selection Masking in other painting modes, Sculpt Mode primarily uses Masks and Face Sets to easily control the mesh visibility and which faces can currently be edited. The exception is the Clipping Region, which can be used in any mode.

The most common shortcuts are `H` to hide the face set under the cursor and `Shift-H` to isolate the face set under the cursor (or show everything).

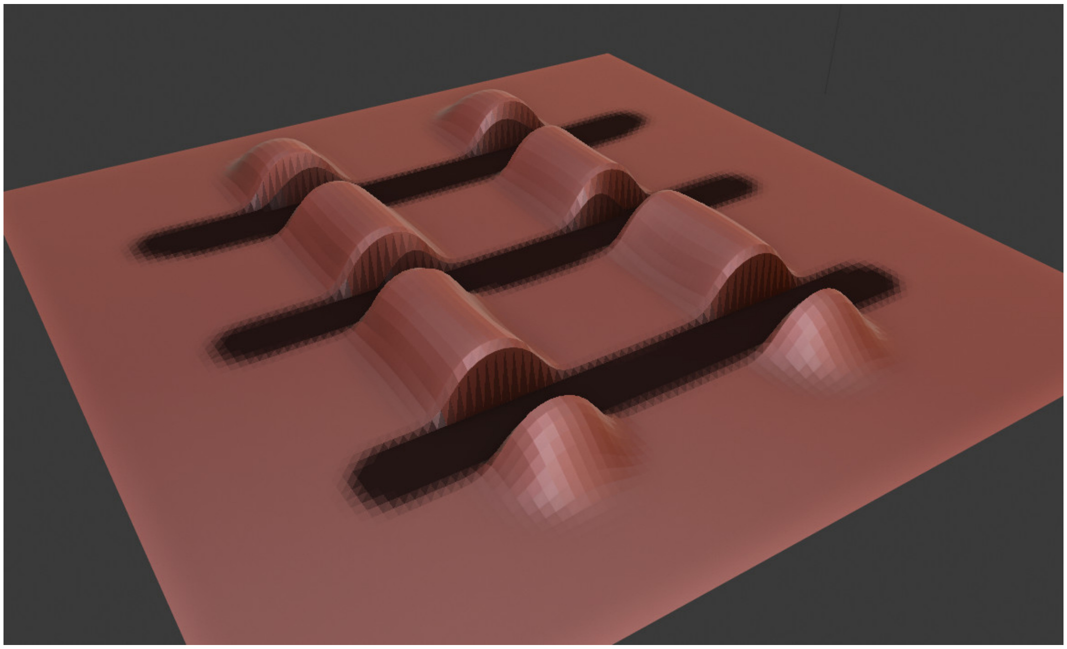
Inverting the visibility and showing all is also available in the `Alt-W` pie menu.

Modifying visibility can also be done via the Hide Gesture Tools.

See also

More information for controlling the visibility at Show & Hide.

Masks



A mask is used to control which vertices of the mesh are influenced by sculpting and painting. The mask can for example be created/edited via the Mask Gesture Tools and Mask by Color tool.

Internally, masks are stored using the `sculpt_mask` Attributes.

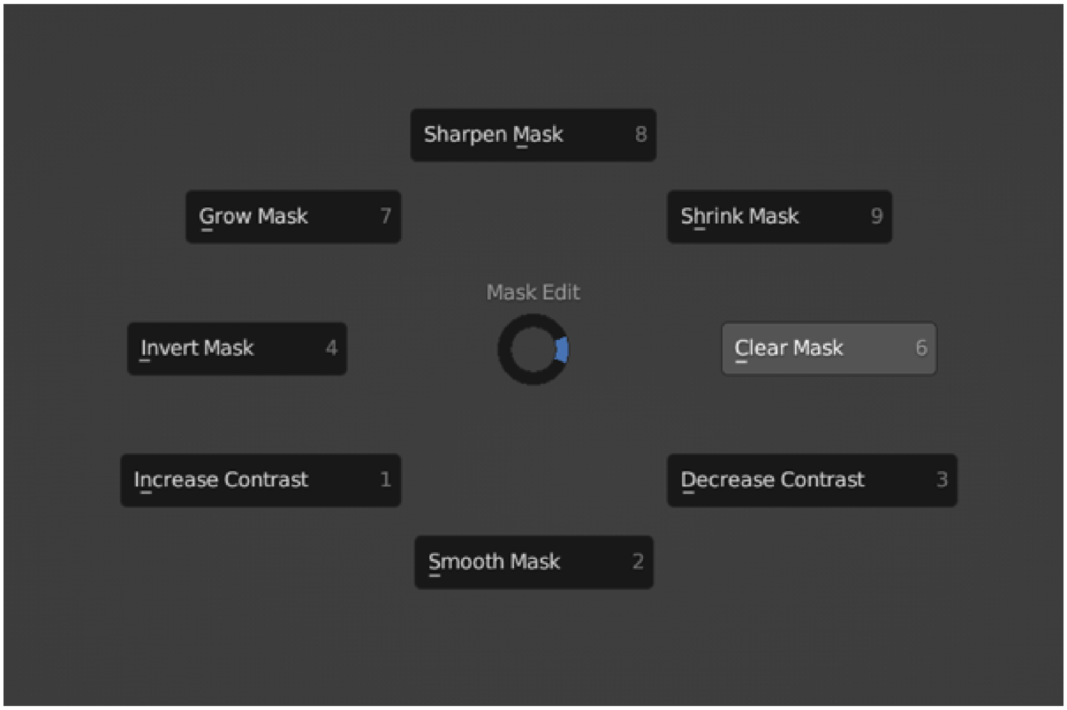
Clear & Invert

Creating masks follows a slightly different mental model than selecting in other modes. For example `Shift-LMB` is used for smoothing instead of adding to a mask.

Masking is also conceptually inverted to selection (i.e. You **cannot** edit masked vertices. But you **can** edit selected vertices).

Instead a mask is typically always added to the current mask with `LMB` and subtracted with `Ctrl-LMB`. So if you wish to edit the masked surfaces, you'll need to use the Invert operator, In the case of masking everything that is visible, the best workflow is to first Clear and then Invert the mask.

Both these operators can be quickly accessed in the `A` pie menu.


Sharpen Mask 8
Grow Mask 7
Shrink Mask 9
Mask Edit
Invert Mask 4
Clear Mask 6
Increase Contrast 1
Decrease Contrast 3
Smooth Mask 2

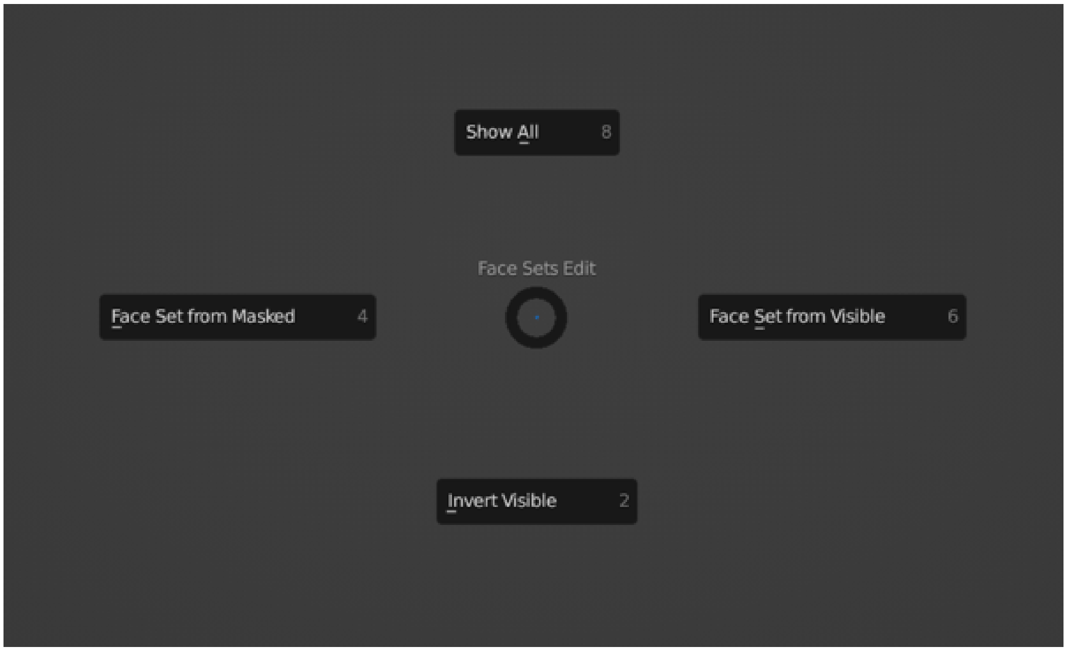
See also

More information about editing and using masks at the Mask Menu

Face Sets

Face sets are used to group your mesh into differently colored faces, which can then be quickly hidden or shown like mentioned above. They can also be used for fast mask creation via the Mask Expand. Face Set Expand is also useful for creating, editing and joining face sets.

More options can be found in the `Alt-W` pie menu.


Show All 8
Face Sets Edit
Face Set from Masked 4
Face Set from Visible 6
Invert Visible 2

Otherwise Face Sets can be created/edited with the Draw Face Sets brush, Mask Gesture Tools. They can also be edited with the Edit Face Set tool.

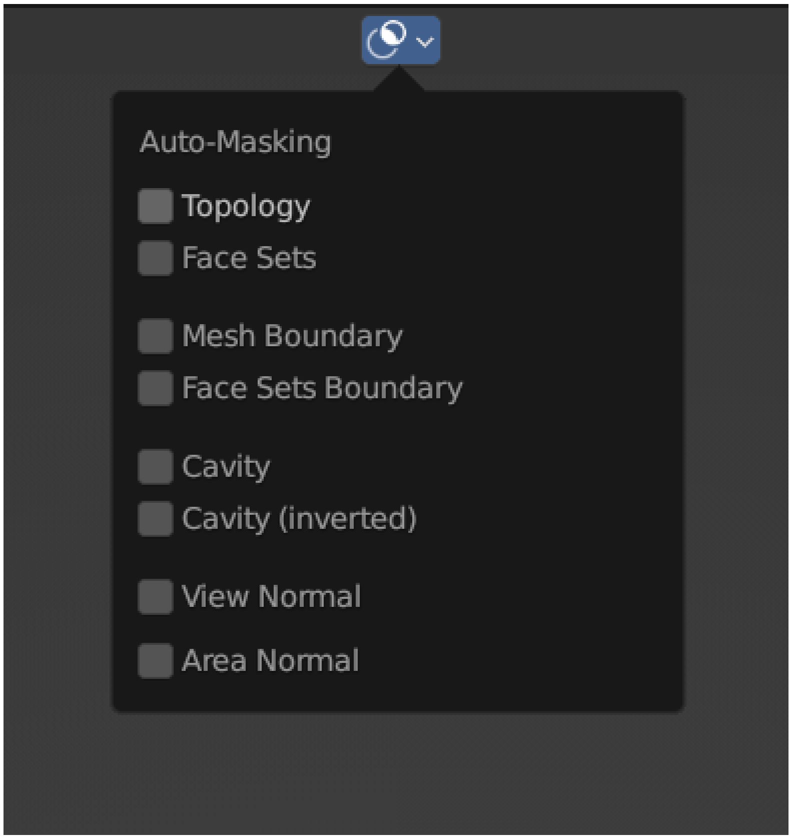
See also

More information about editing and using face sets at the Face Sets Menu

Internally, face sets are stored using the `sculpt_face_set` Attributes.

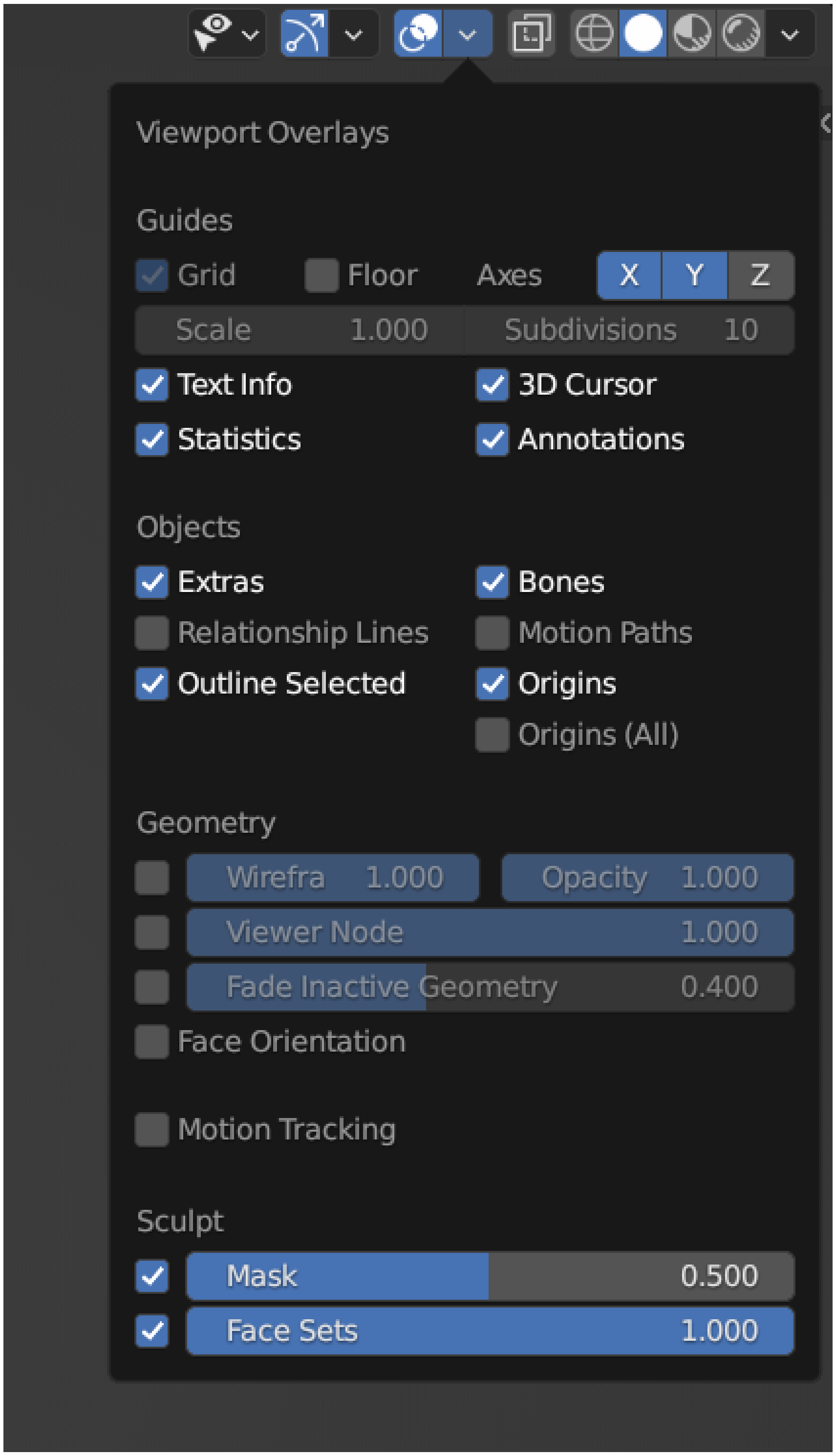
Auto-Masking

Auto-Masking is also a fast way of only editing specific geometry without having to manually create a new mask or hide geometry. This feature is especially useful in combination with face sets.


Auto-Masking
☐ Topology
☐ Face Sets
☐ Mesh Boundary
☐ Face Sets Boundary
☐ Cavity
☐ Cavity (inverted)
☐ View Normal
☐ Area Normal

Display Settings

The mask and face sets display can be toggled and adjusted in the Display Settings.



Viewport Overlays

Guides

☑ Grid ☐ Floor Axes X Y Z

Scale 1.000 Subdivisions 10

☑ Text Info ☑ 3D Cursor

☑ Statistics ☑ Annotations

Objects

☑ Extras ☑ Bones

☐ Relationship Lines ☐ Motion Paths

☑ Outline Selected ☑ Origins

☐ Origins (All)

Geometry

☐ Wirefra 1.000 Opacity 1.000

☐ Viewer Node 1.000

☐ Fade Inactive Geometry 0.400

☐ Face Orientation

☐ Motion Tracking

Sculpt

☑ Mask 0.500

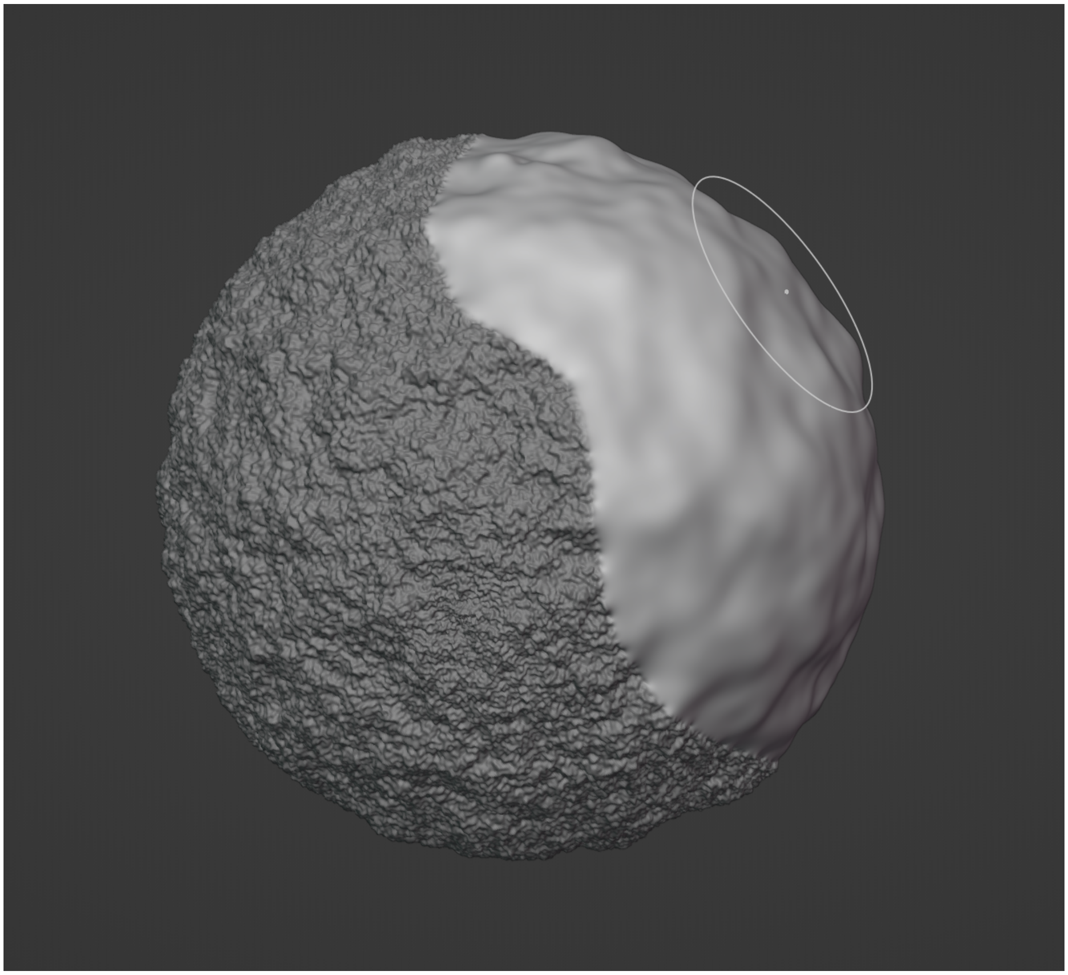
☑ Face Sets 1.000

Note

When Xray shading is enabled, masks and face sets will not be displayed.

Filters

Filters are tools which provide an alternative way of sculpting, because they do not rely on a brush radius. Instead they will affect any vertices that are visible and not masked.



The strength is controlled by click & dragging from left to right. The position of the cursor can be used to only affect specific areas, if auto-masking is used.

Many of the same brush types are also available as a filter type. This way much of the mesh can simultaneously be smoothed, colored or have some cloth simulation applied.

Tip

A common example for using the Mesh Filter is to smooth everything after increasing the resolution with the Voxel Remesher or Dyntopo.

See also

More information at Mesh Filter, Cloth Filter, Color Filter and Mask Filters.

Transforming

Transform tools to move, rotate and scale are also available in Sculpt Mode, but with an important difference to other modes. Sculpt Mode uses its own pivot point, which can be manually positioned `Shift-RMB` or automatically positioned with Mask Expand. This ensures that the pivot point can be more freely placed and always moves with the transformed geometry.

Optionally instead of keeping the transform tools active, you can enable the viewport gizmos to have access to the gizmo at all times.

Note

The gizmo can in some cases block areas from being sculpted on. In that case move the pivot point somewhere else to be able to click on the desired surface.

Apart from the transform tools there are also special brushes to move, rotate and scale the topology like Pose, Boundary and Elastic Deform.

Painting

Sculpt Mode also allows painting your geometry via Color Attributes such as Vertex Colors. This ensures that the most common actions related to the sculpting workflow are contained in the same mode, to avoid unnecessary mode switching.

Other sculpt mode features such as face sets, masking and filters can also be used with painting tools.

The painting functionality in Sculpt Mode is limited to a Paint and Smear brush, as well as a Color Filter and Mask by Color tool.

Just like any other brush, `Shift` can be used to smooth. In the case of painting brushes it will blur the colors within the brush radius instead.

Note

Once any painting tool is executed, the viewport color shading is switched to “Attribute”. This ensures that color attributes are shown on all objects once painting is needed.

Working with Multiple Objects

Unlike Edit Mode, there is no multi-object editing supported for Sculpt Mode. Since sculpting often involves editing many separate objects, it is recommended to use the shortcut `Alt Q` while pointing at other objects, for Switching Objects quickly.

The advantage of using multiple objects is that each can have its own origin and modifiers. Splitting the geometry among multiple objects can also improve the sculpt mode performance. Alternatively objects can also be joined so there is no need to switch objects.

In the case that Face Sets were already used, joining objects or creating new geometry in Edit Mode will automatically assign new Face Sets. This makes it immediately possible to target each new geometry, for example via auto-masking. If no Face Sets are created, use the Initialize Face Sets operator to create them.

Face Sets and Masked geometry can also be extracted via Expand Mask or sliced into a new object via Mask Slice.

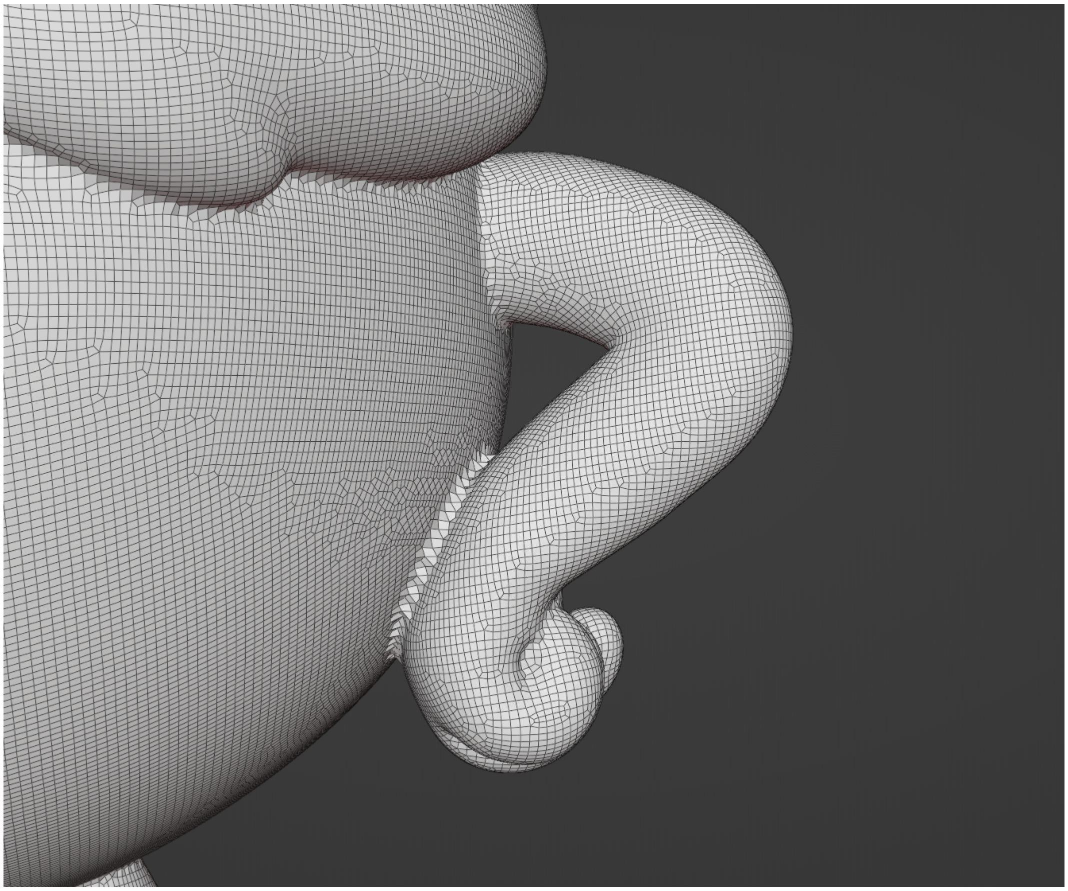
Adaptive Resolution

In order for sculpting to give accurate and predictable results, Blender needs enough geometry. Instead of starting out with a highly subdivided mesh, add geometry dynamically by using either of the following adaptive sculpting methods.

Voxel Remesher

“Voxel remeshing” rebuilds the geometry with a perfectly even distributed topology. Depending on the set voxel size, this will lead to a lower or higher resolution.

This technique is especially useful to block out the initial shape of an object. It also has the advantage of removing any overlapping geometry and creating a manifold volume as a result.



Any currently used mask, face sets and color attributes will be re-projected on the remeshed result. Reaching high vertex counts should still be achievable with this technique, depending on the used hardware.

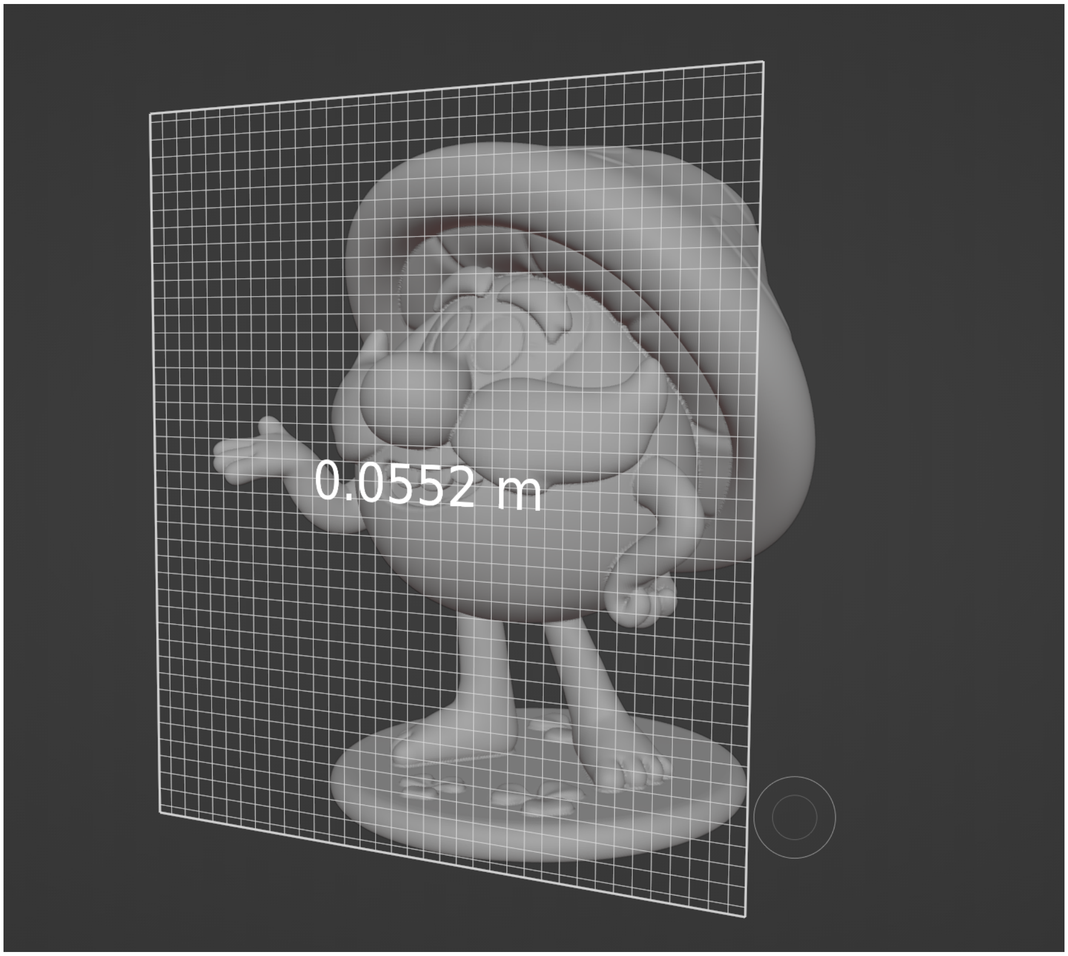
Note

This technique will not work on objects that do not have an enclosed volume. Make sure to fill any holes in the mesh before remeshing. Or avoid any holes in the mesh/volume that are larger than the defined voxel size.

Tip

If in doubt, you can fill all holes in edit mode or by using the [Mask Slice and Fill Holes] operation to fill all holes in the mesh. If nothing is masked, it only fills any holes.

To more easily access this feature, use the shortcuts `R` to define the resolution, and `Ctrl-R` to execute the remeshing.

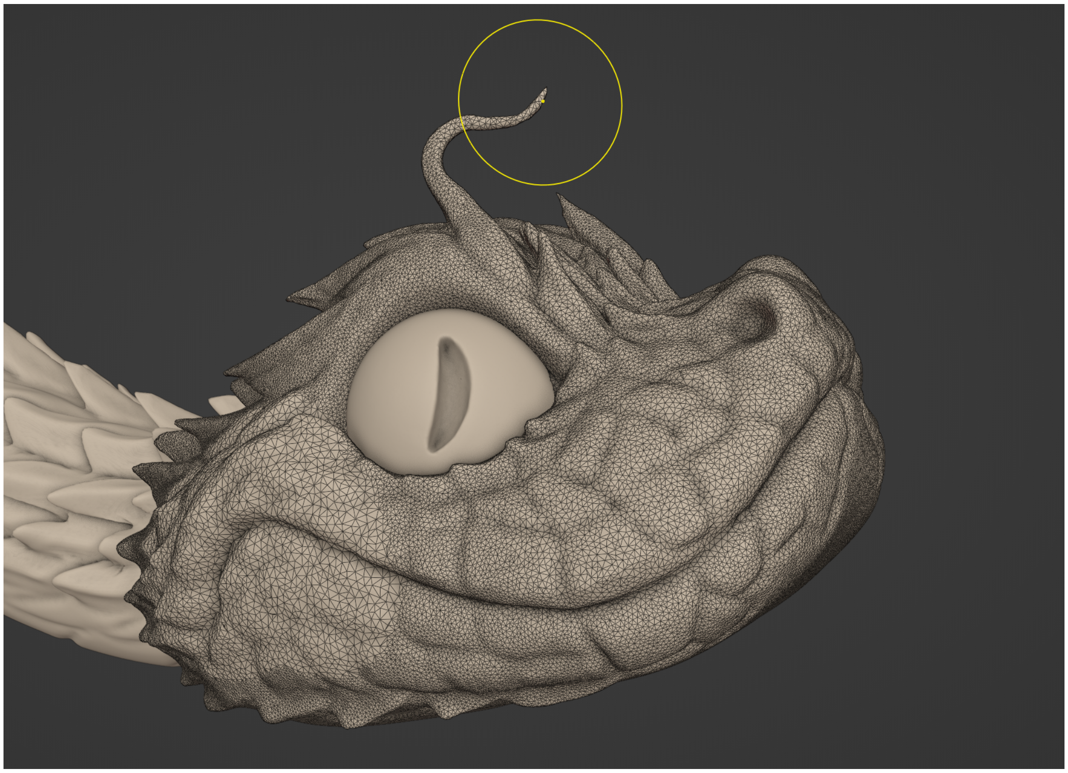


See also

More information at [Remesh].

Dyntopo

Dynamic topology (aka Dyntopo) is a dynamic tessellation sculpting method that automatically adds and removes topology under the brush.



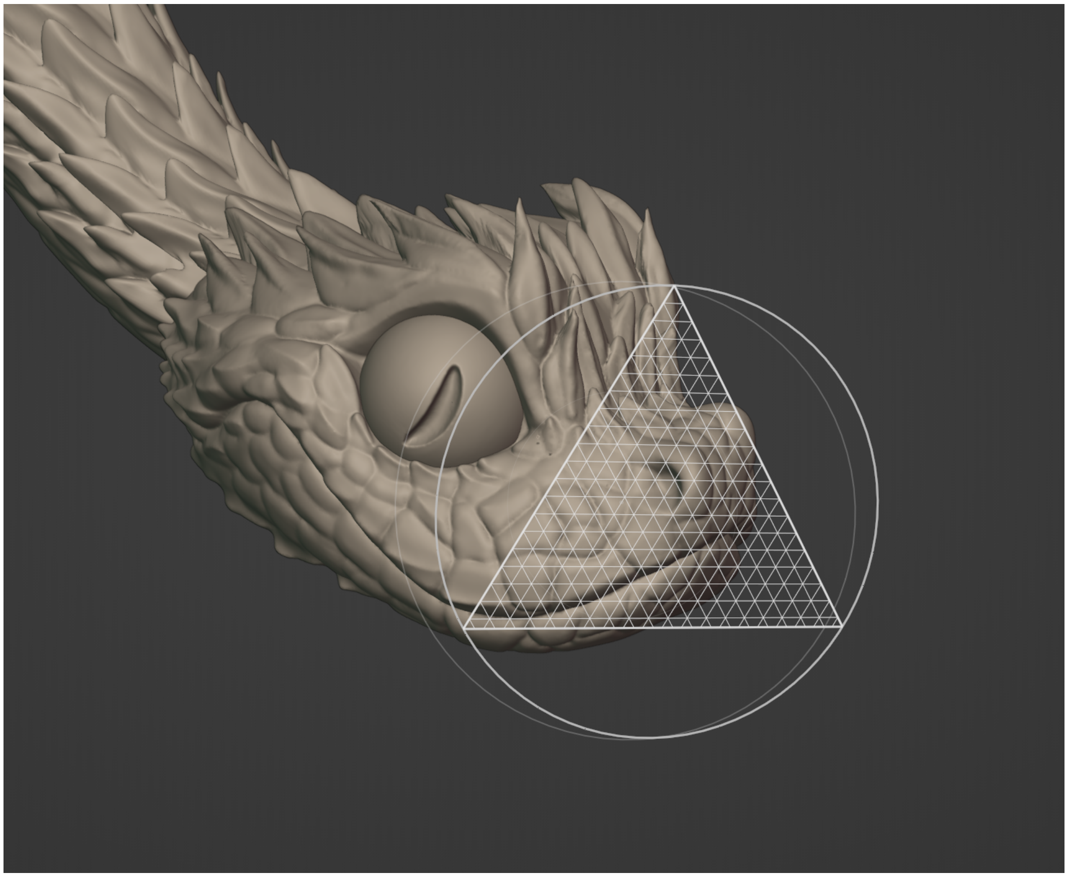
Unlike the Voxel Remesher, this makes it possible to sculpt complex shapes without thinking about the resolution or topology. It also allows to define a different resolution wherever necessary. Much more complex base mesh sculpting is especially useful with this technique.

The disadvantages of this technique are a slower performance and limited support for some sculpt mode features. Custom attributes like Color Attributes, UV Maps and Face Sets are also lost or corrupted when using Dyntopo.

This feature shares the same shortcuts with voxel remeshing when enabled. Use `R` to define the resolution and `Ctrl-R` to flood fill the resolution (if Constant Detail is used).

Note

Because Dyntopo and the Voxel Remesher are mutually exclusive and cannot be used at the same time, both use the same shortcut to define the remeshing resolution.



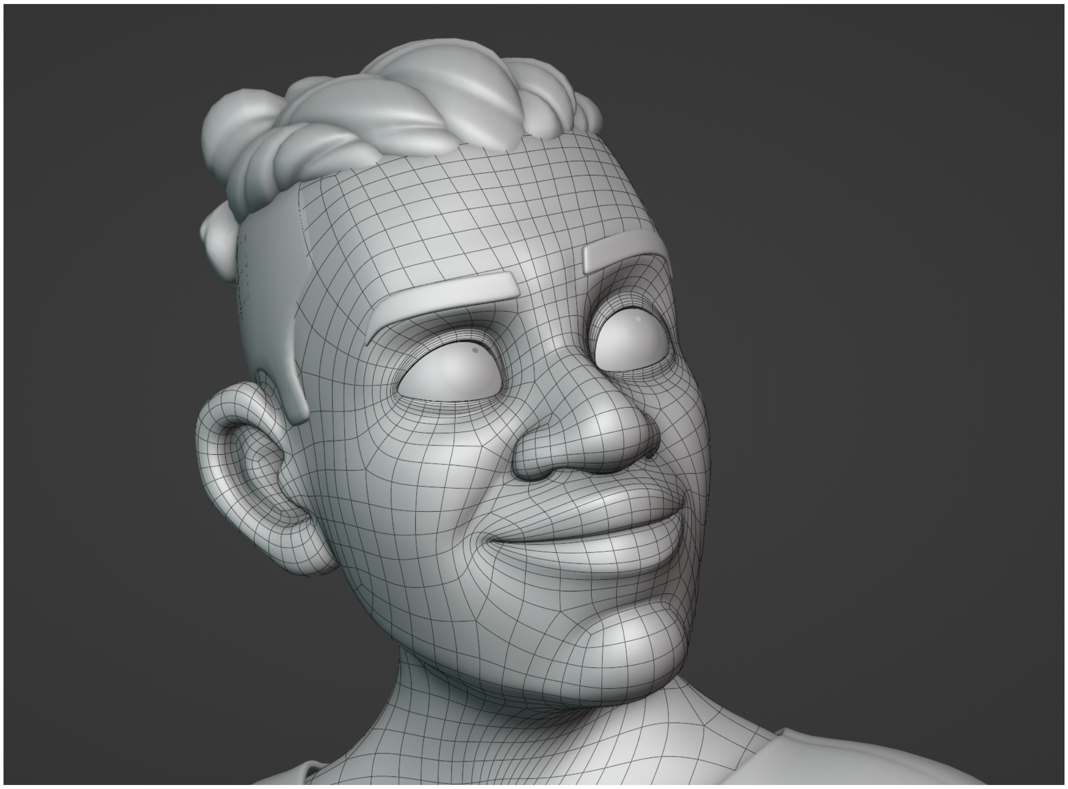
Brushes like Density, Snake Hook and Clay Strips work especially well with this feature.

See also

More information at Dyntopo.

Multiresolution

The Multiresolution Modifier can be used for subdivision based sculpting. This means the object will be subdivided, similar to the Subdivision Surface Modifier, only that the subdivisions can be freely sculpted for very high resolution detailing.



Note

For this technique it is highly recommended to use on a clean topology base mesh. This means the base mesh should be only made of quads and avoid non-manifold faces, as well as poles with two connected edges. More information at Quad Remeshing for an automatic retopology method.

This technique has the advantage of sculpting with multiple resolutions, meaning you have the ability to sculpt on any level of subdivision. This allows to add a much higher resolution of details for rendering and sculpting, while displaying lower resolutions for better viewport performance. It also allows sculpting on lower resolutions any time for broader changes.

As an example, you can sculpt general proportions in subdivision level 1, add high resolution details in level 4 and switch back to subdivision 1 to correct the shape further.

The disadvantages are that you may end up with some mesh distortions because the topology is not dynamic like voxel remeshing and dyntopo. The topology should also not be changed once already subdivided, since any edits to the base mesh will result in corrupted subdivision details.

Tip

Pay attention to the topology that you sculpt and how much it gets stretched. If more resolution is needed you can always subdivide another time, but there will be worse performance and slower level switching once more than 5 subdivisions are used. Alternatively use the Slide Relax brush to slide topology to where it is needed.

Additional brushes like the Multires Eraser and Multires Smear are recommended for adjustments.

Here are general shortcuts to use the feature.

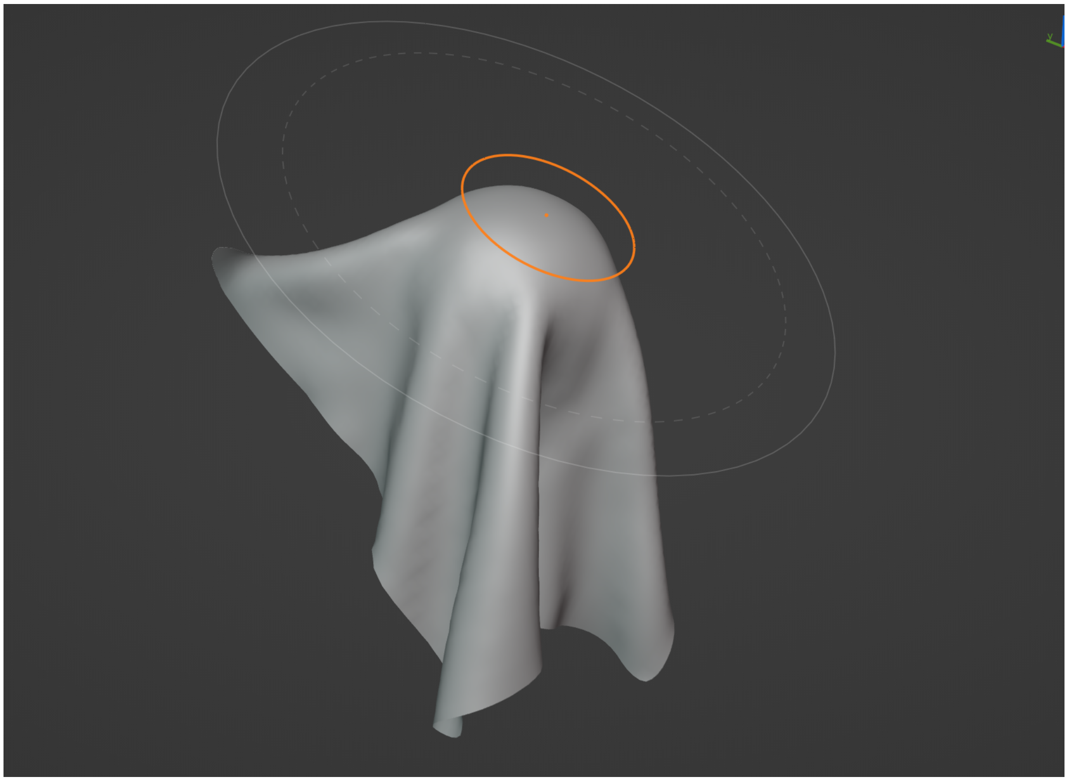
- Step up one multires level `Alt-2`
- Step down one multires level `Alt-1`
- Set multires level / Create multires modifier `Ctrl-0` to `Ctrl-5`

See also

More information at Multiresolution Modifier.

Cloth Sculpting

Instead of sculpting cloth manually or creating complex physics simulation setups, there are various tools directly in sculpt mode that offer a simplified Cloth Physics Simulation.



This has various advantages but is especially useful for base mesh creation and larger clothing folds and draping. Detailing is possible, but the slower performance on high resolution meshes and simplified cloth physics might not lead to desirable results.

The resolution of the topology is mainly responsible for the size of the folds and detail level of the simulation. So an optimal and evenly distributed topology is important.

Many sculpting features are supported, so for example Masked vertices are pinned in the simulation. Another example is with auto-masked face set